Case Study - Automatic Tuning of Machine Learning for Robust and Reliable Anomaly Detection

Abstract

- Automatic tuning of machine learning algorithms is a promising approach for improving the robustness and reliability of anomaly detection.
- By leveraging optimization techniques such as hyperparameter tuning, feature selection, and model selection, automatic tuning can help to identify the most effective algorithms and settings for a given dataset and task.
- One key benefit of automatic tuning is its ability to reduce the need for manual intervention and domain expertise, which can be time-consuming and error-prone.
- However, automatic tuning also presents challenges such as computational complexity, overfitting, and generalization, which must be carefully addressed to ensure the validity and effectiveness of the resulting models.
- Some promising techniques for automatic tuning of machine learning algorithms for anomaly detection include Bayesian optimization, evolutionary algorithms, and reinforcement learning.
- Overall, automatic tuning has the potential to significantly enhance the performance and applicability of anomaly detection in various domains, including cybersecurity, fraud detection, and predictive maintenance.

Introduction

Goals

- Finding anomalies using unsupervised algorithms.
- Automatic tuning of hyperparameters of the algorithms
- Find best hyperparameters using flaml tuning for each 40 training datasets.
- Compare the AUC scores of 10 test datasets using default and suggested parameters.

Anomaly Detection

- Anomalies are unusual or rare events or patterns in a dataset.
- It is most often used when it is easy to collect a large amount of known-normal examples where anomalous data is rare and difficult to find.
- Machine learning algorithms can improve the accuracy and effectiveness of predictive models by detecting anomalies.
- It can help solving problems, e.g. bank fraud, medical problems, structural defects, malfunctioning equipment etc.

Challenges in Anomaly Detection

Lack of labelled data

Imbalanced datasets

Noise

Why Unsupervised Algorithms?

Unsupervised

- detect anomalies without the need for any prior knowledge of what an anomaly looks like
- can be used to detect wide range of anomalies
- minimizes the number of false positives, saves times

Supervised

- Only limited to detect the type of anomalies for which labeled data is available
- unexpected events difficult to label in a supervised manner
- labeled data may not be available or difficult to obtain

Hyperparameter Tuning

- process of finding the best combination of hyperparameters for a machine learning algorithm
- determine the performance of models
- can help reduce the time and computational resources required to train a model
- can help prevent overfitting and increase the model's generalization performance by right set of hyperparameters

Challenges in Hyperparameter Tuning

Large number of possible combinations in general

difficulty in optimizing without knowing labelled data

Interactions between hyperparameters

Need of domain expertise

Solution Framework

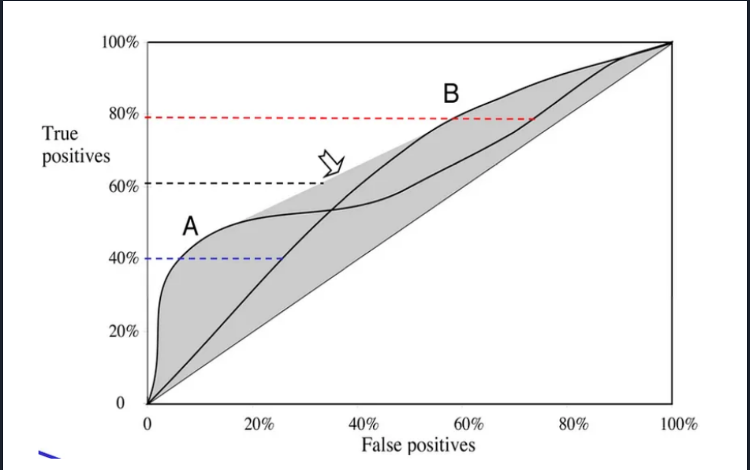
Evaluation metric

- AUC (Area Under the curve)

  TPR = True Positives / All Positives
  FPR = False Positives / All negatives

- AUC ranges in value from 0 to 1.

  AUC of 0.0 = Model predictions are 100% wrong
  AUC of 1.0 = Model predictions are 100% correct



Source: Maryam Shoaran

Hyperparameter tuning using Flaml

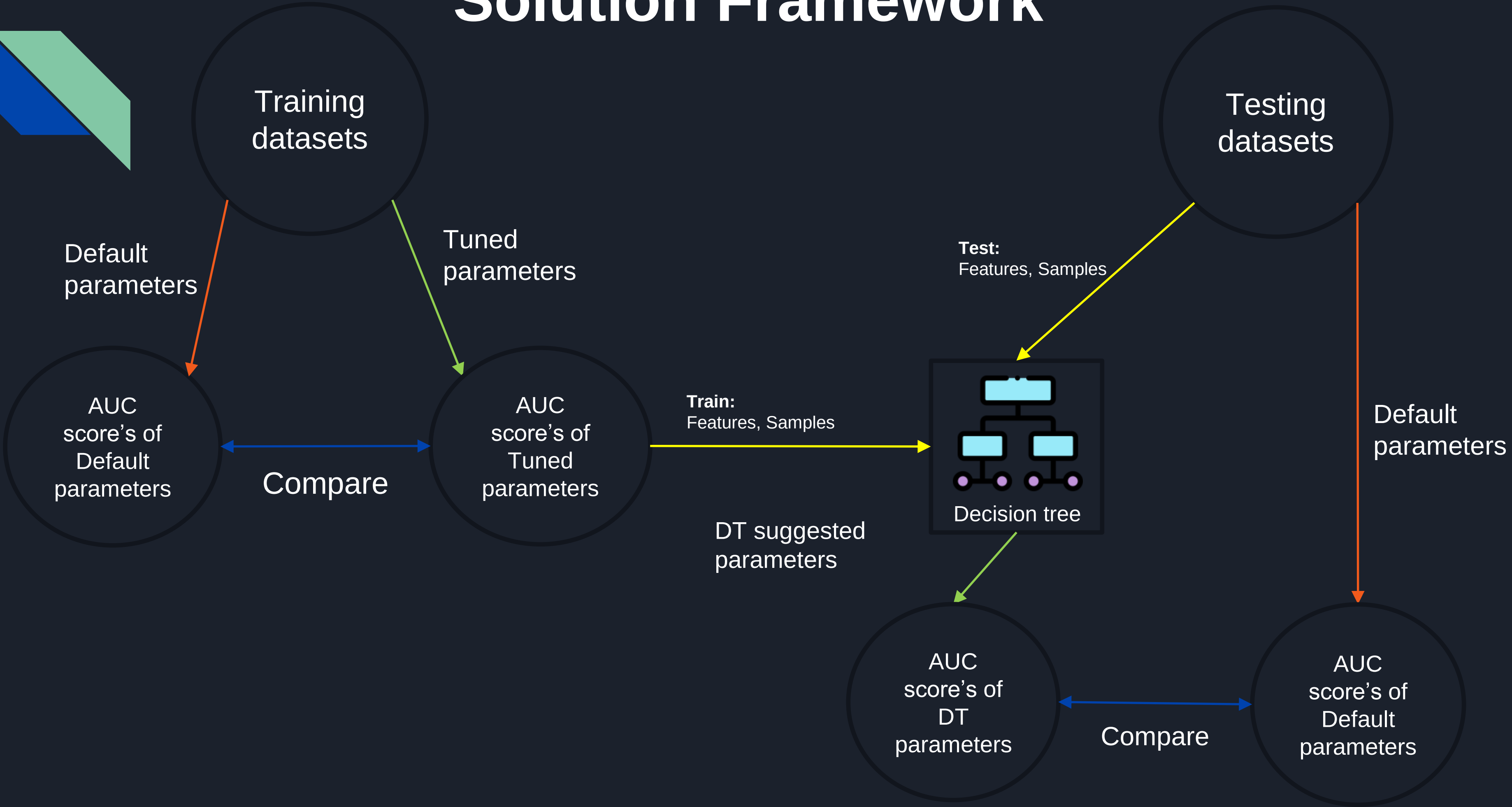
- A Python Library for Automated Machine Learning & Tuning
- FLAML offers a fast auto-tuning tool powered by a novel cost-effective tuning approach
- Evaluation function to tune hyper parameters of an algorithm for optimized AUC score for the given input data.

Find Quality Model at Your Fingertips

Easy to Customize or Extend

Tune It Fast, Tune It As You Like

Solution Framework



Algorithms

Autoencoder (Under)

Umid Kumar Dey

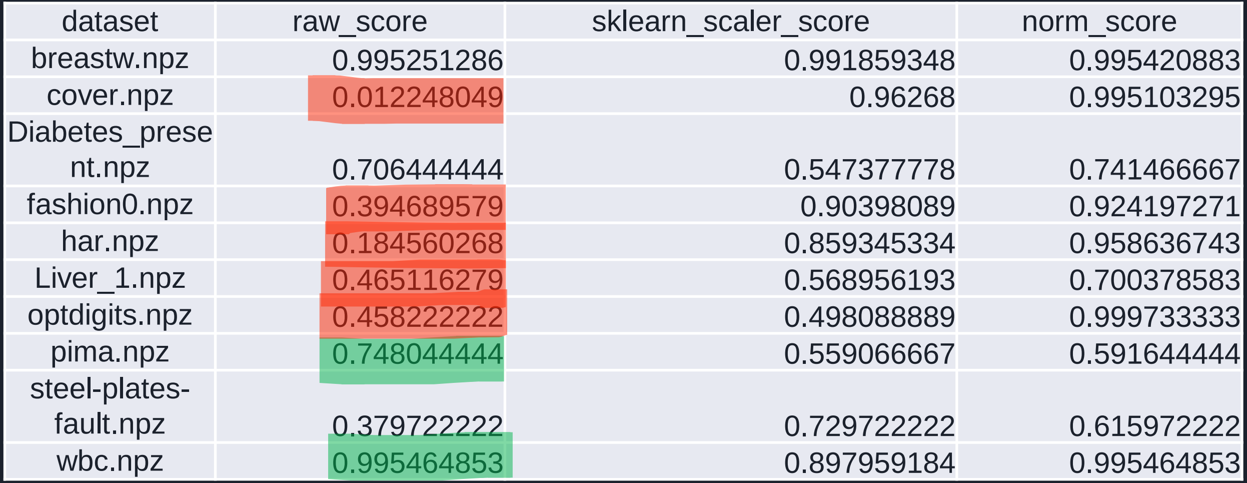
To-do: AutoEncoder overview

- Anomaly detection is a crucial task in many applications, including fraud detection, cyber-security, and predictive maintenance.
- Autoencoders are a type of neural network that have emerged as a powerful tool for anomaly detection due to their ability to learn complex patterns in data.
- In simple terms, autoencoders have neurons that compress input data and then decompress it to reconstruct it.
- By training the autoencoder on normal, non-anomalous data, it can learn to recognize the underlying structure and patterns in that data.
- Once trained, the autoencoder can be used to detect anomalies by comparing the reconstruction error of new, unseen data with that of the training data.
- Anomalies are likely to have a higher reconstruction error than normal data, as the autoencoder struggles to reconstruct them accurately.

To-do: Interesting observations (Umid)

Since we don‘t exactly create the conditions and rules for the neural network in the autoencoder algorithm and instead rely on Keras to do that job, it fundamentally boils down to the data it is being fed; with that in mind, some observations:

- Raw unprocessed data will almost certainly not give as good results as processed ones



dataset	raw_score	sklearn_scaler_score	norm_score
breastw.npz	0.995251286	0.991859348	0.995420883
cover.npz	0.012248049	0.96268	0.995103295
Diabetes_present.npz	0.706444444	0.547377778	0.741466667
fashion0.npz	0.394689579	0.90398089	0.924197271
har.npz	0.184560268	0.859345334	0.958636743
Liver_1.npz	0.465116279	0.568956193	0.700378583
optdigits.npz	0.458222222	0.498088889	0.999733333
pima.npz	0.748044444	0.559066667	0.591644444
steel-plates-fault.npz	0.379722222	0.729722222	0.615972222
wbc.npz	0.995464853	0.897959184	0.995464853

- For the other 40 datasets, only 9 of them had better results with raw input

To-do: Interesting observations (Umid)

Normalised data [(x – x.min)/(x.max – x.min)]

- Higher kurtosis difference between test and train dataset leads to AUC score less than 0.8 more often - 9/19 times in the training datasets (test set results were inconclusive)
- Higher kurtosis difference also had higher scores than 0.8 but only 5/21 times – 4 of those 5 datasets, however, had 10 or more features.
- More features doesn‘t corelate to better scores

Sklearn scaled data

- Interestingly, here, the effect of kurtosis difference is exactly the opposite; i.e. higher kurtosis difference led to better score (>0.8) – 9/28 times – but could still be deemed inconclusive since 19 other better 0.8+ scores had lower kurtosis difference
- However, only 4/12 times there were worse scores with a higher kurtosis difference, making it difficult to make conclusions about distribution effect on accuracy
- More features gives better scores (22/32 times with 0.8+ for >= 10 features)

PCA and Autoencoder (Over)

Khandakaer Abir Hossain

PCA & Autoencoder

Principal Component Analysis

- linear method that reduces dimensionality of the data while preserving its most important features
- simpler and faster technique than autoencoder
- may perform worse than autoencoder on complex and non-linear datasets

Autoencoder

- non-linear technique that learns a non-linear mapping between the input and the latent space.
- capture more complex relationships between the features than PCA
- autoencoder may be computationally expensive compared to PCA.

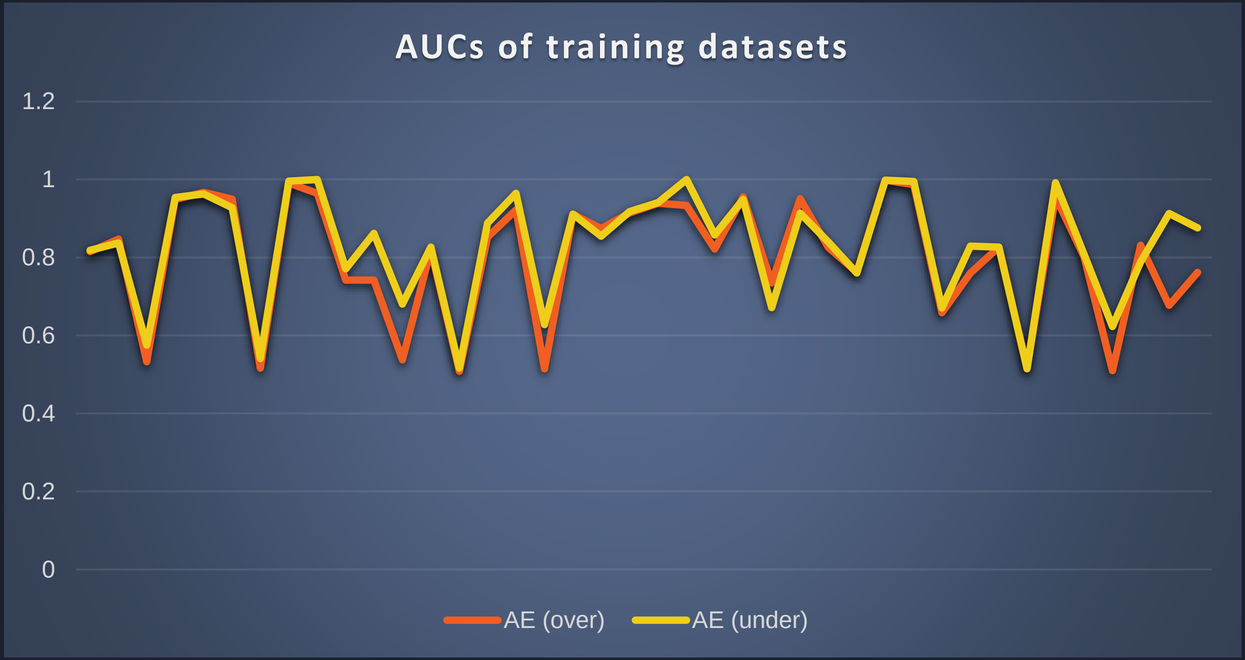
PCA vs Autoencoder

Datasets	PCA AUCs	Autoencoder AUCs
Diabetes_Present	**0.6896**	0.6530
Har	**0.8867**	0.8863
Cover	**0.9432**	0.9426
Breastw	**0.9922**	0.9913
Fashion0	0.9024	**0.9032**
Prima	**0.7194**	0.6722
Liver_1	**0.5614**	0.5608
Wbc	**0.9932**	0.9841
Steel-plates-fault	0.7176	**0.7243**
optdigits	**0.5320**	0.5301

Comparison of Autoencoder (Over) and (Under)

- Autoencoder (Over): Latency size is fixed (10* number of features)
- Autoencoder (Under): Latency size tuned (less than number of features)
- Optimizers: 'adam', 'sgd','rmsprop', 'adagrad', 'adadelta’
- Loss functions: binary_crossentropy, mean_squared_error, mean_absolute_error

Autoencoder (Over) vs (Under) with best found params



Autoencoder (Over) vs (Under) with suggested params

Datasets	Autoencoder (Under)	Autoencoder (Over)
Diabetes_Present	0.5474	**0.7064**
Har	0.8593	**0.8858**
Cover	**0.9627**	0.9429
Breastw	0.9919	0.9919
Fashion0	0.9040	**0.9228**
Prima	0.5591	**0.6078**
Liver_1	0.5690	**0.5722**
Wbc	0.8980	**0.9846**
Steel-plates-fault	**0.7297**	0.7271
optdigits	0.4981	**0.5245**

ABOD and VAE

Nishat Tasnim Ahmed Meem

Angle-Based Outlier Detection (ABOD)

- A geometric approach to detect anomalies by measuring the angles between a set of any three datapoints.
- The variances in the angles of outliers are usually low compared to the variances of the normal datapoints.
- Variance values less than a certain threshold can be marked as potential anomalies.

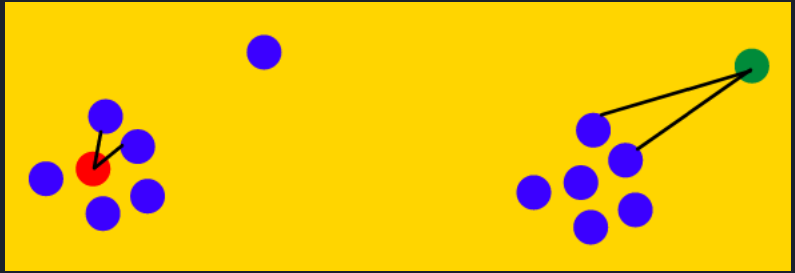


Image source - https://blog.paperspace.com/outlier-detection-with-abod/

ABOD – Hyperparameter tuning

- n_neighbors (**default=10**) - Number of neighbors to use by default for k neighbors queries.
- Mean of the all n_neighbors found from tuning 40 training datasets - **24**

Autoencoder

- Autoencoder has an encoder that maps the input data to a lower-dimensional latent space representation.
- The decoder tries to map the latent space representation back to the original data.

Variational Autoencoder

- The encoder of VAE outputs the mean and the standard deviation for each latent variable.
- The latent vector is sampled from this mean and standard deviation which is then fed to the decoder to reconstruct the input.

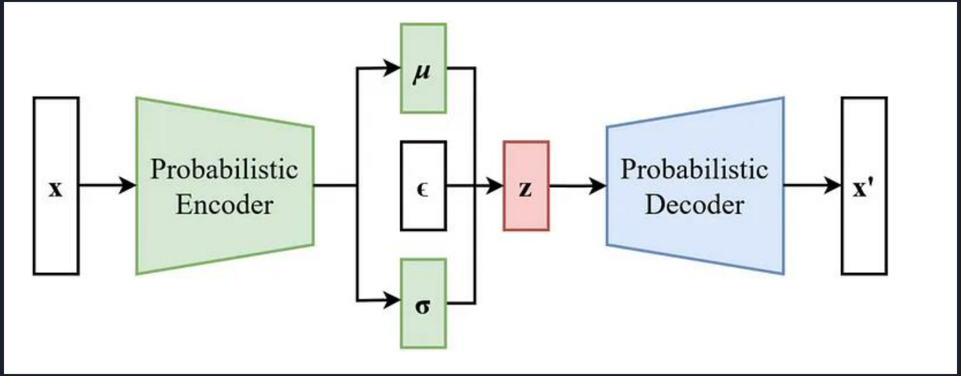


Image - Variational Autoencoder
source: wikipedia — https://en.wikipedia.org/wiki/Variational_autoencoder

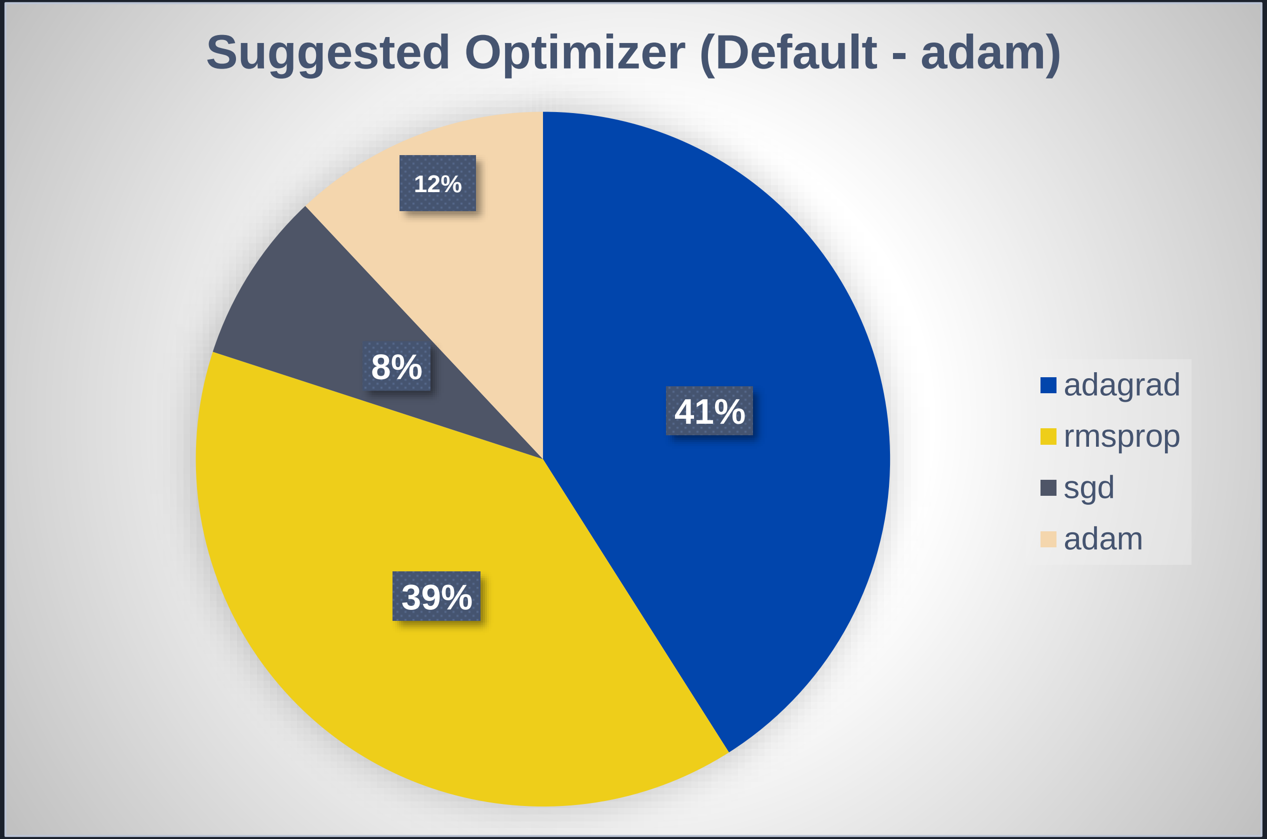
VAE for Anomaly Detection

- Reconstruction error is the difference between the original input and the output of the decoder.
- VAE learns to produce less reconstruction error for data that are similar to the data it was trained on.
- An input that generates an extreme reconstruction error is likely to be an anomaly.
- The training loss of VAE is the sum reconstruction the loss and the similarity loss.
- The similarity loss is the KL divergence between the latent space distribution and standard gaussian (zero mean and unit variance).

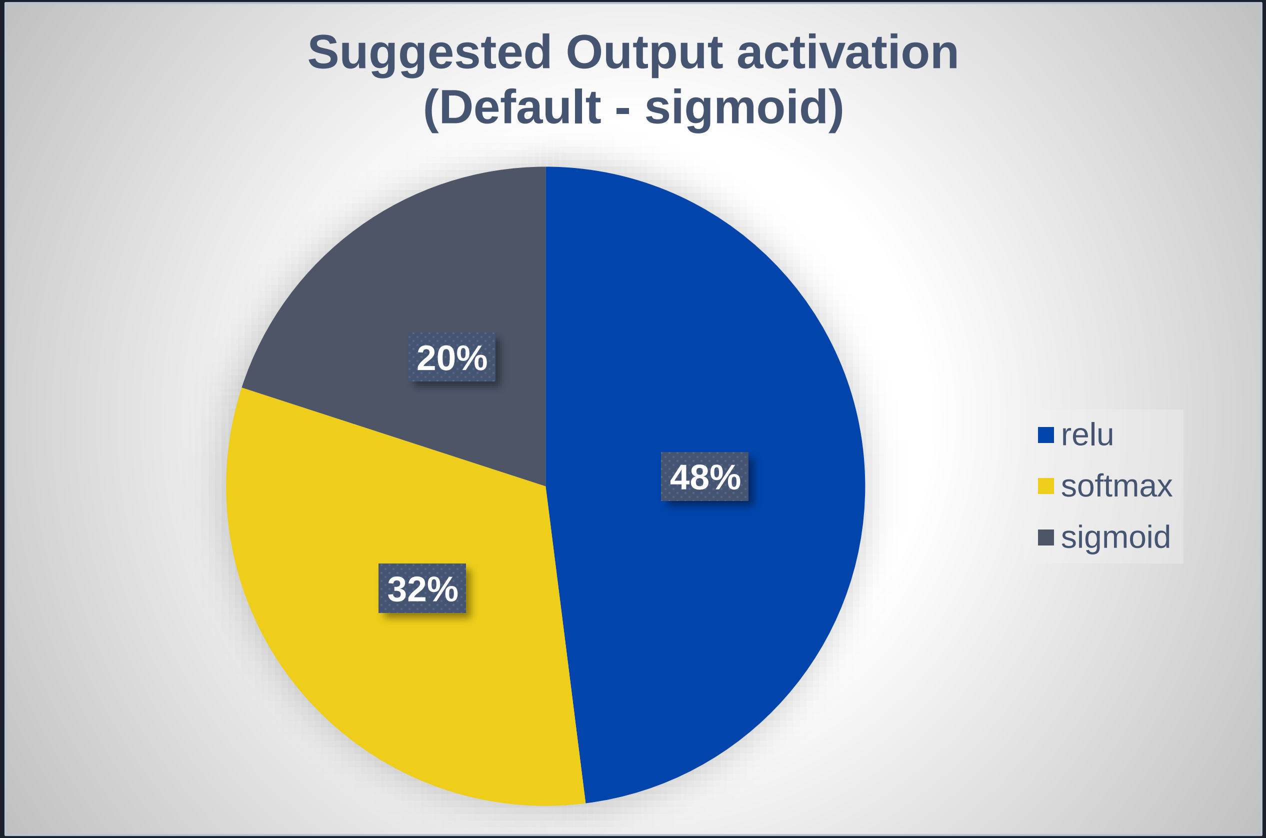
VAE – Hyperparameter tuning

Hyperparameters	Default value	Suggested hyperparameters by flaml
Encoder Neurons	[128, 64, 32]	[104, 24, 18]
Decoder Neurons	[32, 64, 128]	[18, 24, 104]
Epochs	100	415
Dropout Rate	0.2	0.72
L2 Regularizer	0.1	0.15

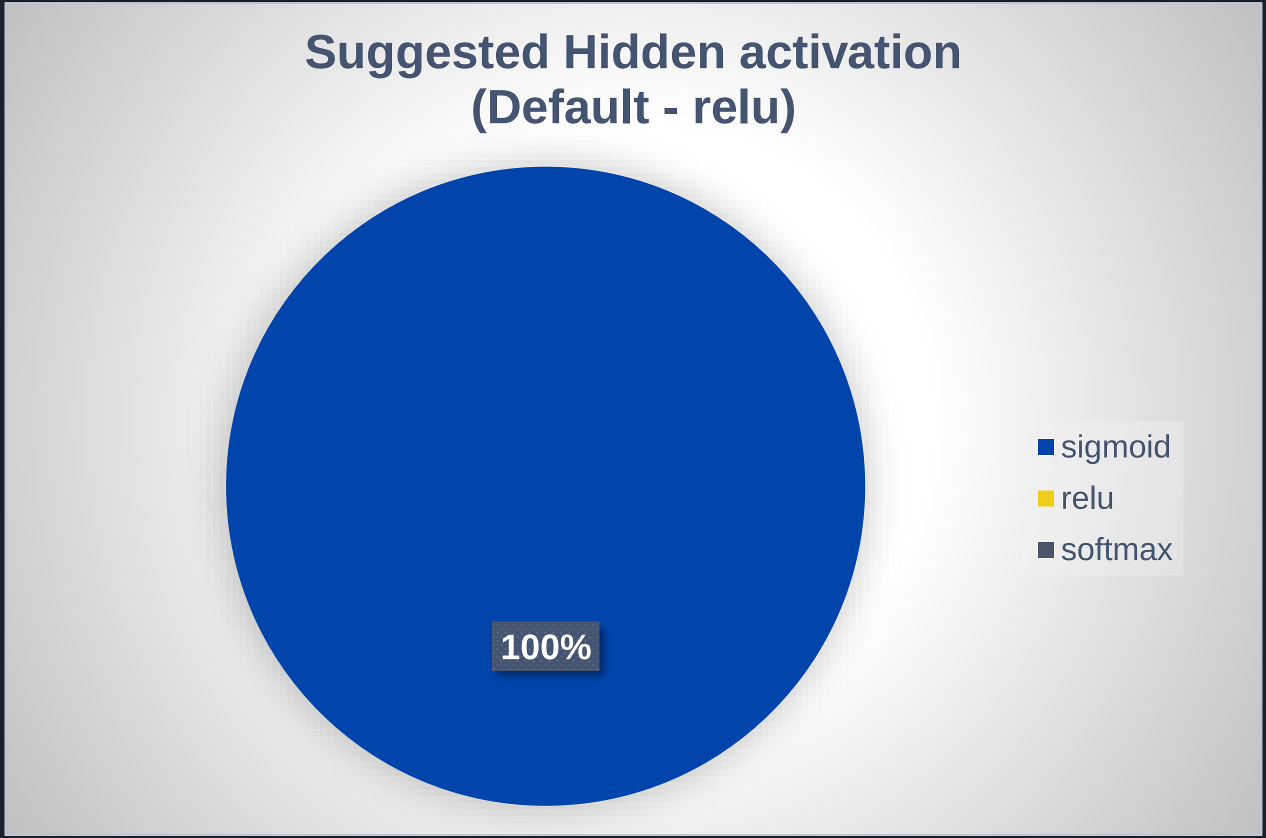
VAE – Hyperparameter tuning



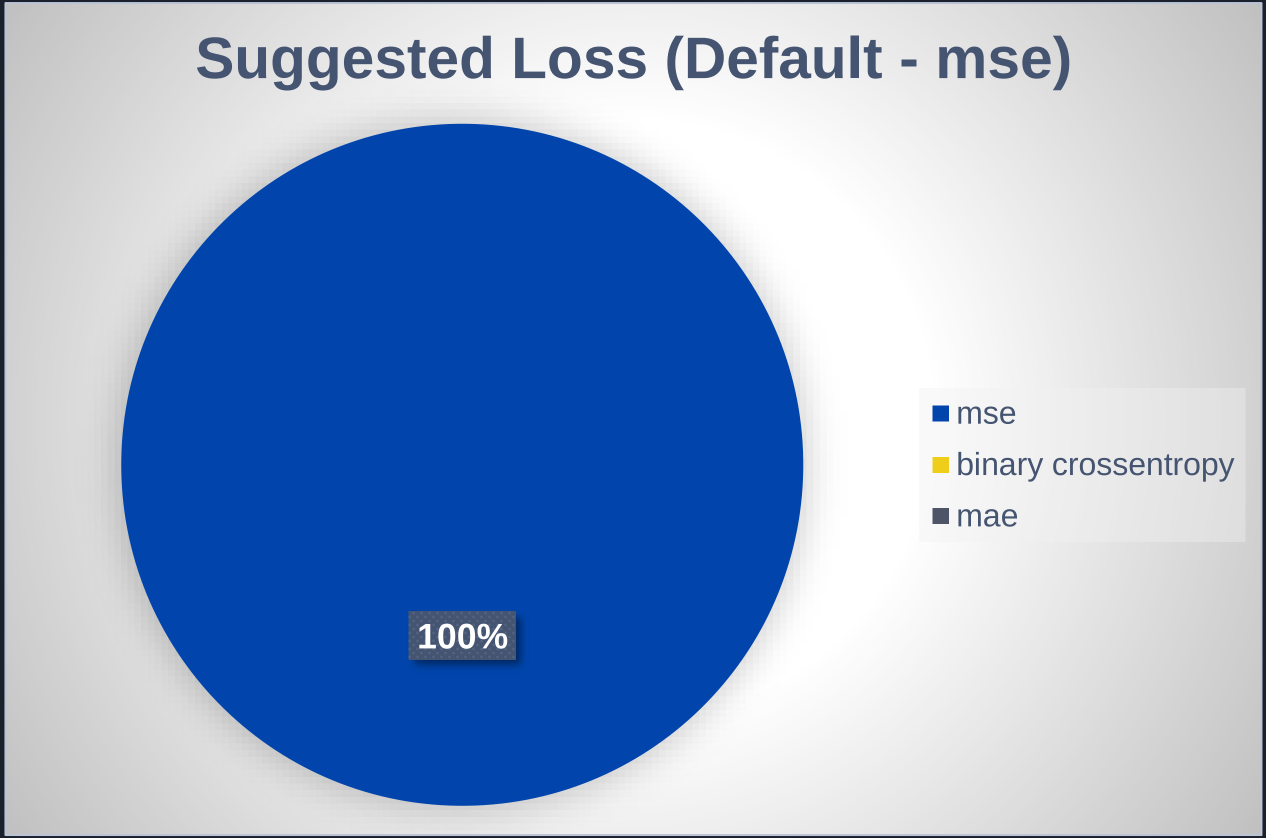
VAE – Hyperparameter tuning



VAE – Hyperparameter tuning



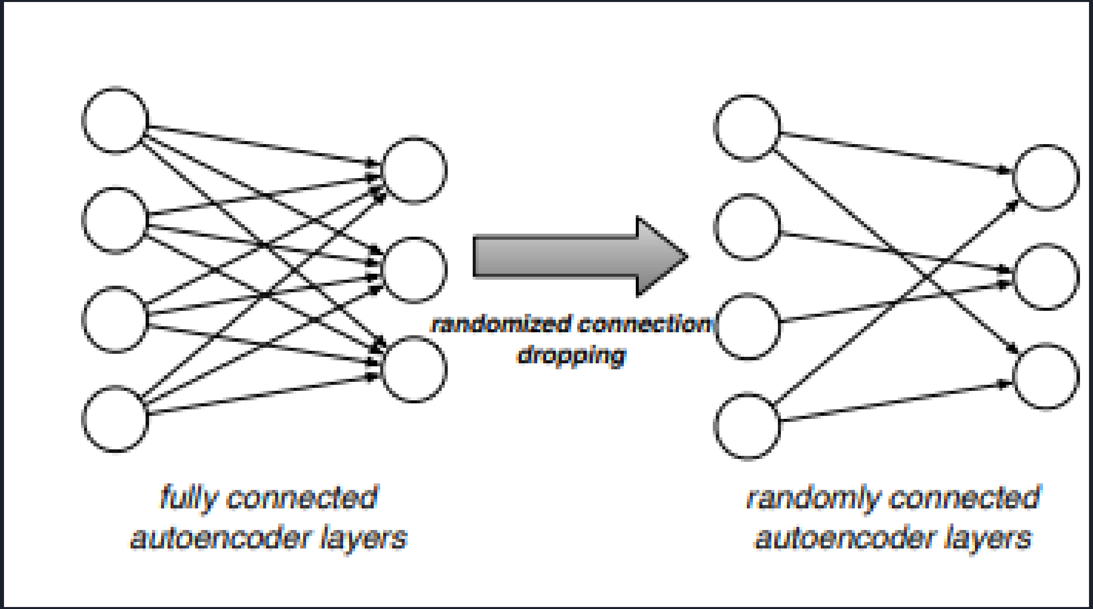
VAE – Hyperparameter tuning



RandNet

By Upanishadh Prabhakar

RandNet - Randomized Neural Network for Outlier Detection


randomized connection dropping
fully connected autoencoder layers
randomly connected autoencoder layers

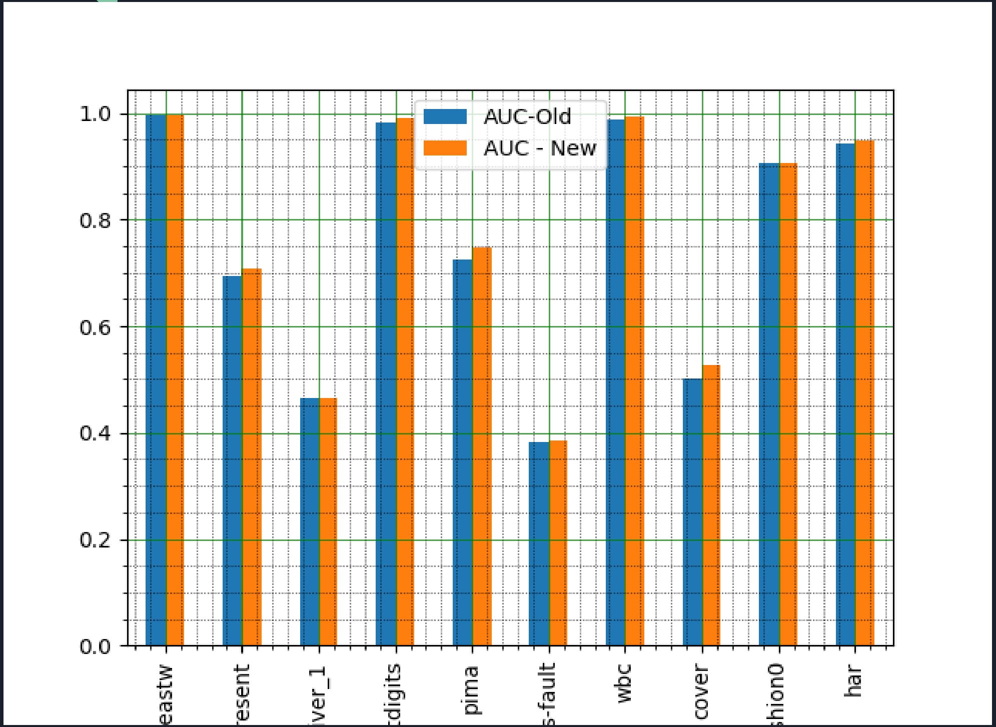
- Diversity in individual components of the ensemble framework is required.
- Same network structure for all ensemble components leads to similar output, which is unhelpful from the perspective of variance reduction.
- Solution : Randomly connected autoencoder models in which some connections are randomly dropped.

Src ; https://saketsathe.net/downloads/autoencode.pdf

RandNet - Randomized Neural Network for Outlier Detection

- Process – L1 and L2 adjacent layers, then L1.L2 possible connections. Here, sample |L1.L2| connections with replacement.
- Sample will have some repeated connections and some missing connections.
- Missed connections - Connections that will be dropped for the next autoencoder ensemble component.

RandNet - Randomized Neural Network for Outlier Detection



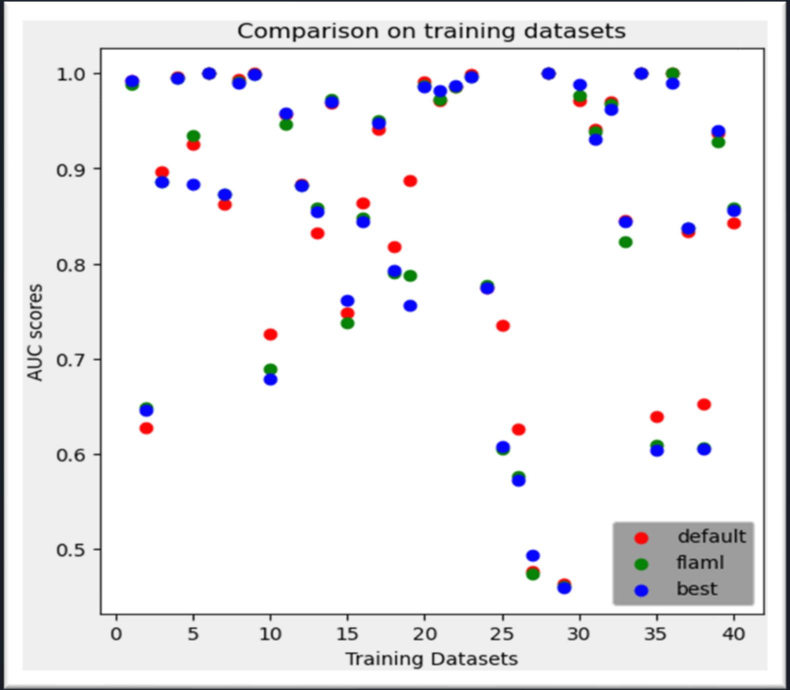
- Hyperparameter – ‘α’ : Structure Parameter – Higher value of α, more nodes, more powerful reconstruction ability.
- Higher α leads to a higher AUC until a certain level, but causes issues in test data because of overfitting.

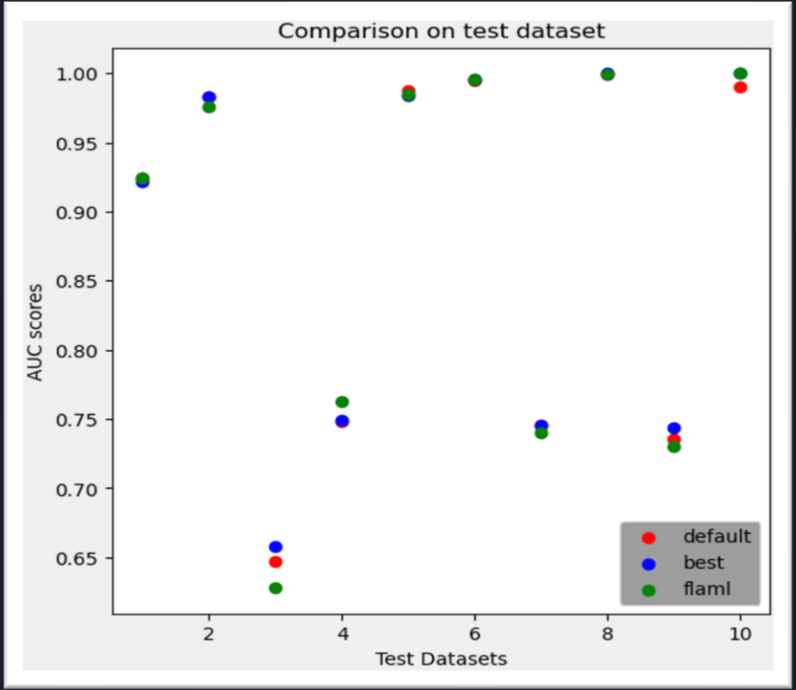
LUNAR

Kartik Kamboj

LUNAR (Learnable Unified Neighbourhood-based Anomaly Ranking)

Lunar is a neural network framework with trainable parameters, which distinguishes it from other local outlier approaches and increases its flexibility and adaptability to a dataset.





- Majority of the training datasets have good AUC scores.
- Some datasets have better AUC scores for best hyperparameters than the default one.
- Highest AUC is 1.0 and lowest is 0.46 in the training dataset.
- Highest AUC is 1.0 and lowest is 0.65 in the test dataset.
- Flaml parameters - WEIGHT as model type, SUBSPACE as negative sampling, 3 n_neighbours, 0.0621 as epsilon, 925 as n epochs, learning rate (lr) value as 0.006 and proportion as 5.
- Best parameters - WEIGHT as model type, UNIFORM as negative sampling, 2 n_neighbours, 0.022 as epsilon, 758 as n epochs, learning rate (lr) value as 0.003 and proportion as 6.
- No change in the model type parameter - 'WEIGHT'.

LODA and DeepSvDD

Sohith Dhavaleswarapu

Algorithm : LODA (Lightweight on-line detector of anomalies)

- "LODA" can identify features in which the scrutinized sample deviates from the majority using their joint probability of data generating process.
- **Algorithm**:
  - Density Estimation (training):

    Input: data samples

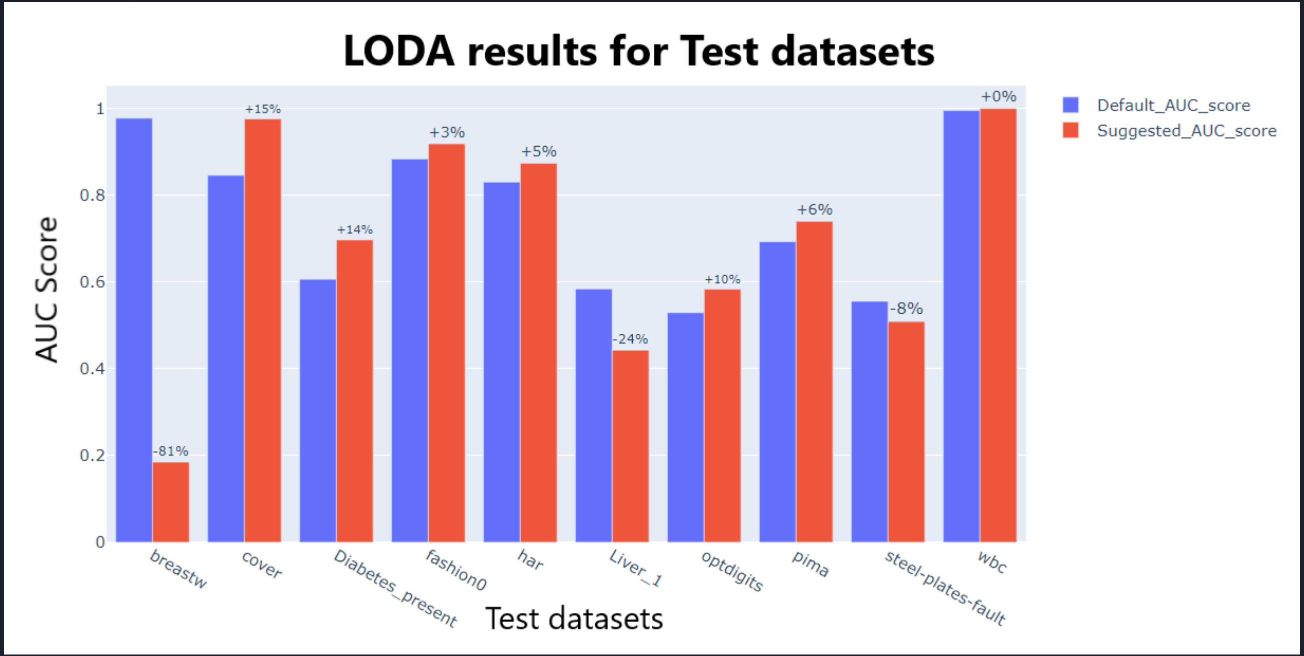
    Output: histograms, projection vectors
  - Classification:

    Input: Set of histograms and projection vectors

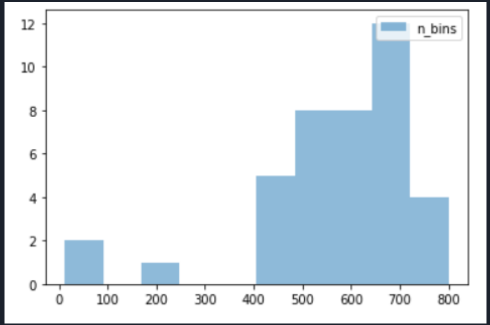
    Output: anomaly value

- **Parameters:**
  - ***n_bins***: The number of bins for the histogram (Default:10).
  - ***n_random_cuts***: The number of random cuts (Default:100).

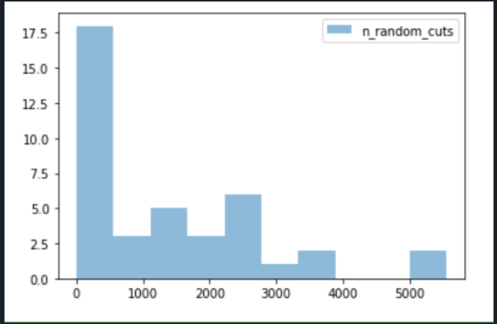
Results



LODA results for Test datasets
AUC Score
Test datasets
Default_AUC_score
Suggested_AUC_score
0
0.2
0.4
0.6
0.8
1
-81%
+15%
+14%
+3%
+5%
-24%
+10%
+6%
-8%
+0%
breastw
cover
Diabetes_present
fashion0
har
Liver_1
optdigits
pima
steel-plates-fault
wbc



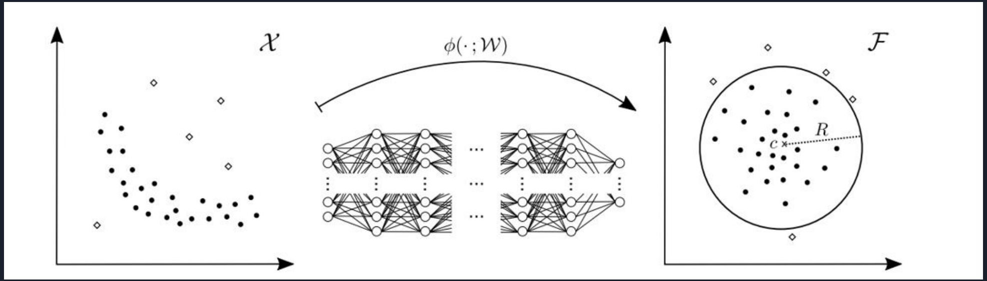
n_bins



n_random_cuts

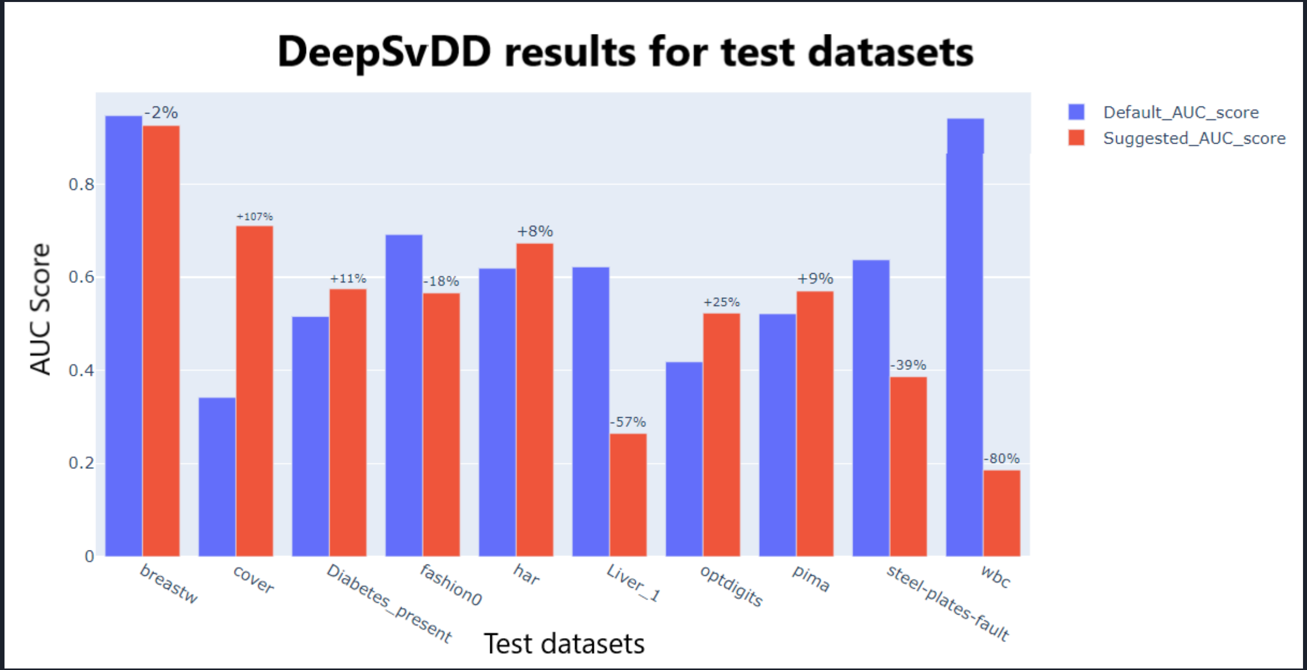
Algorithm : DeepSVDD (Deep One-Class Classification for outlier detection)

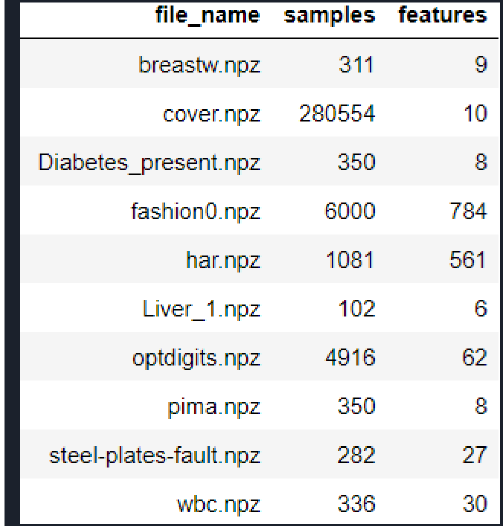
- “DeepSVDD” trains a neural network while minimizing the volume of a hypersphere that encloses the network representations of the data, forcing the network to extract the common factors of variation.



- **Parameters**: c, use_ae, hidden_neurons , ***hidden_activation*** , ***output_activation***, optimizer, ***epochs***, batch_size, ***dropout_rate***, ***l2_regularizer***, validation_size, ***preprocessing***, verbose, random_state, contamination.

Results





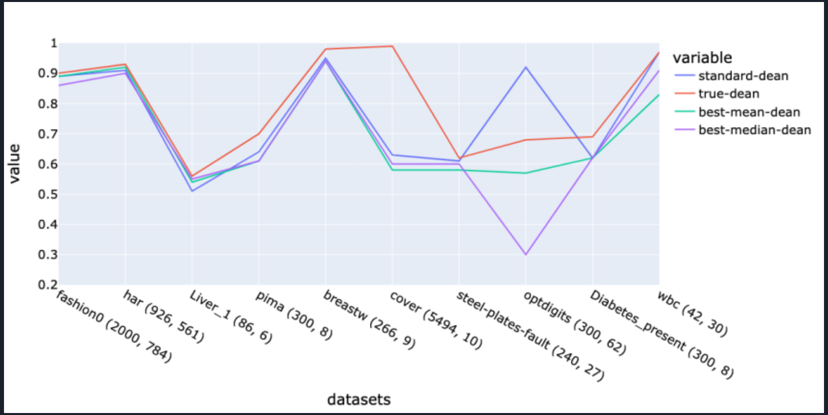
file_name	samples	features
breastw.npz	311	9
cover.npz	280554	10
Diabetes_present.npz	350	8
fashion0.npz	6000	784
har.npz	1081	561
Liver_1.npz	102	6
optdigits.npz	4916	62
pima.npz	350	8
steel-plates-fault.npz	282	27
wbc.npz	336	30

Ensemble models for Anomaly Detection

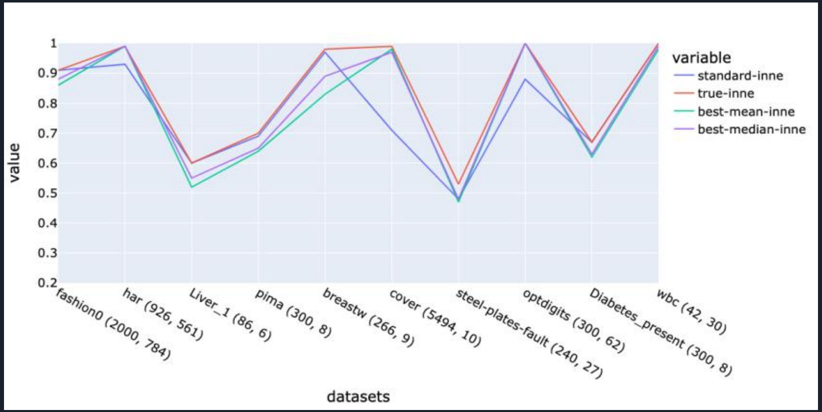
Imene KOLLI

DEAN vs. iNNE

Model	DEAN	iNNE
Architecture		
Based on		
Hyperparameters		
Trained to		

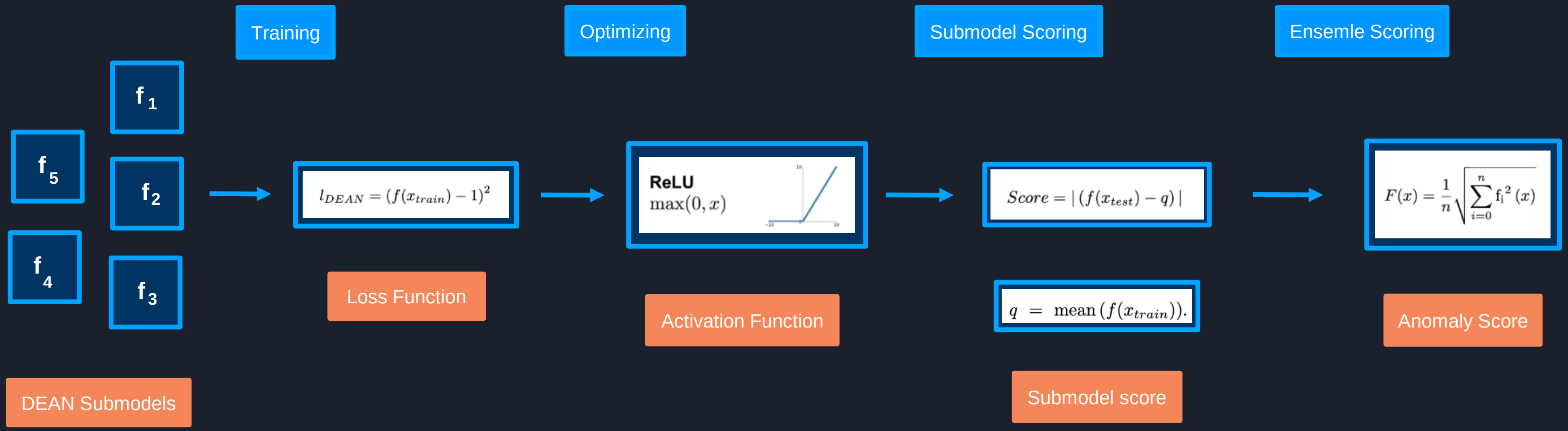


DEAN



iNNE

Deep Ensemble Anomaly Detection - DEAN



$$l_{DEAN} = (f(x_{train}) - 1)^2$$

$$\max(0, x)$$

$$Score = |\,(f(x_{test}) - q)\,|$$

$$q \;=\; \text{mean}\,(f(x_{train})).$$

$$F(x) = \frac{1}{n}\sqrt{\sum_{i=0}^{n} \mathrm{f_i}^2\,(x)}$$

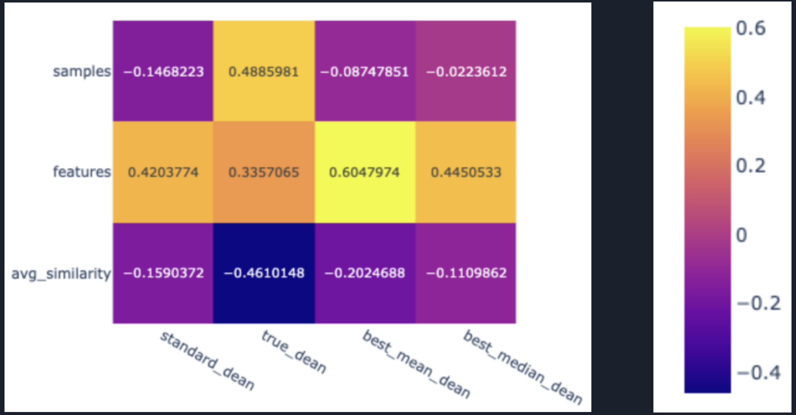
Correlation and the AUC score - DEAN

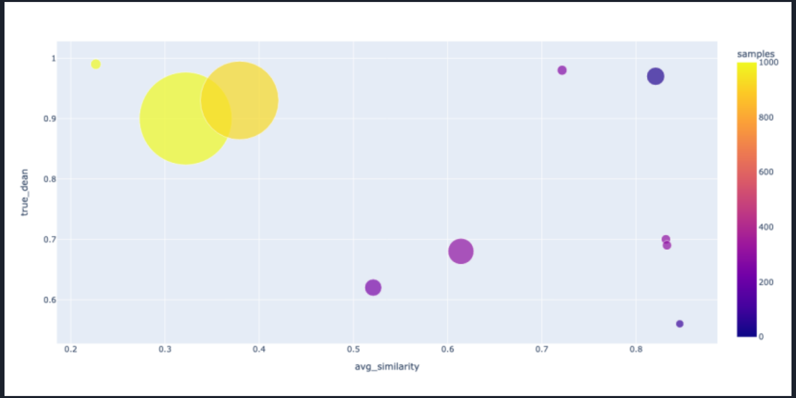
1. DEAN

- The correlation between DEAN's results with different parameters correlates **negatively** with the similarity between the Training and Testing datasets.

- The number of features in the dataset and the AUC score results are **positively** correlated

# Datasets	2	1	5	**2**
Samples				**low**
Features				**low**
Similarity				**high**
AUC score				**High**





The size of the markers represents the number of features

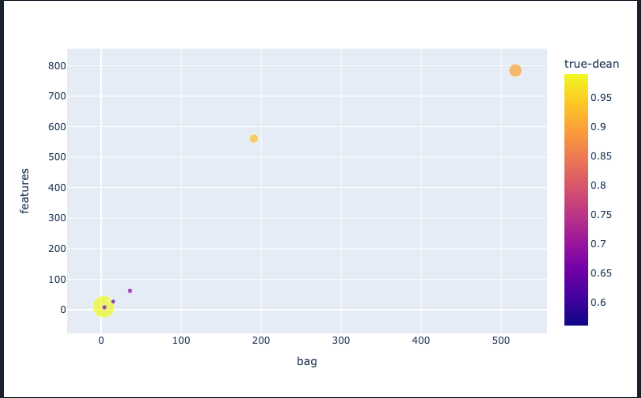
DEAN - Hyperparameters and AUC scores

1. DEAN

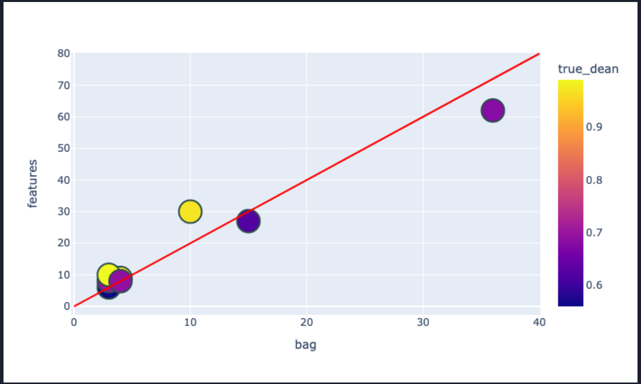
   - For the Low dimensional datasets, the bag size of the model effects the true AUC score

   - If the bag size is more than **50%** of the total number of features in the dataset, the AUC score is **Low** (below the red line)

Datasets	Mean BAG		Standard Bag		True Bag
Bag size	**-50%**	**+50%**	**-50%**	**+50%**	**-50%**
wbc	-	0.83	0.97	-	0.97
breastw	-	0.94	-	0.95	0.98
cover	-	0.58	-	0.63	0.99
har	0.92	-	0.91	-	0.93



The size of the markers represents the sample size



Normalizing Flows

Shubham Gupta

Normalizing flows – Basic Concept

- Model the probability distribution of the data

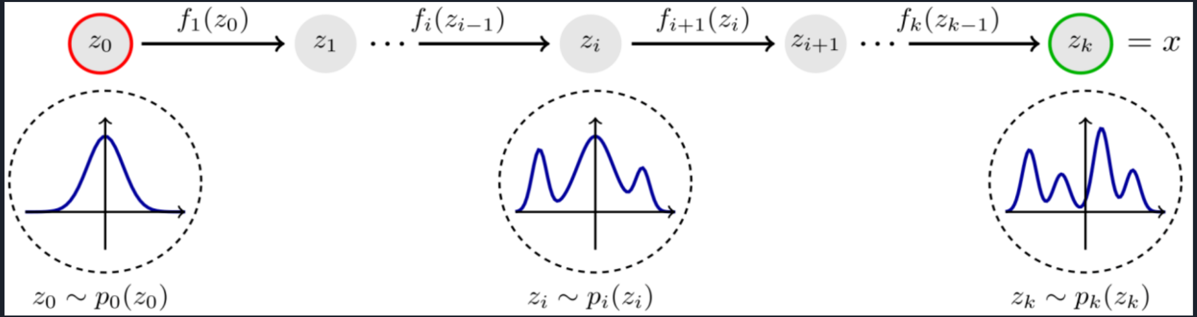


Figure 1 : Transforming a simple distribution to a complex one [1].

- Mapping with invertible transformations

$$z \sim p_\theta(z) \ = \ N(z; 0, I)$$

$$x = \ f_\theta(z) = f_k \circ \ldots \circ \ f_2 \ \circ \ f_1(z), \text{ each } f_i \text{ is invertible}$$

- Change of variable
- Probability density function property
- Log loss function

Normalizing flows for anomaly detection

- **Problem** : Computationally intensive
- **Solution** : Directly for anomaly detection, and do not maintain invertibility

Translation

$$1 = \int p(x)\,dx = \int p(x+\alpha)\,dx = \int p(x+\alpha)\,d(x+\alpha)$$

Scaling

$$1 = \int p(x)\,dx = \int p(\beta \cdot x)\,dx = \frac{1}{\beta}\int p(\beta \cdot x)\,d(\beta \cdot x)$$

Splitting

$$1 = \int p(x)\,dx = \gamma \cdot \int p(x)\,dx + (1-\gamma)\cdot \int p(x)\,dx, \forall\, 0 \leq \gamma \leq 1$$

Mixture Layers

$$1 = \int p(x)\,dx^d = \int p(A \cdot x)\frac{1}{|A|}\,dx^d$$

Non-linearity

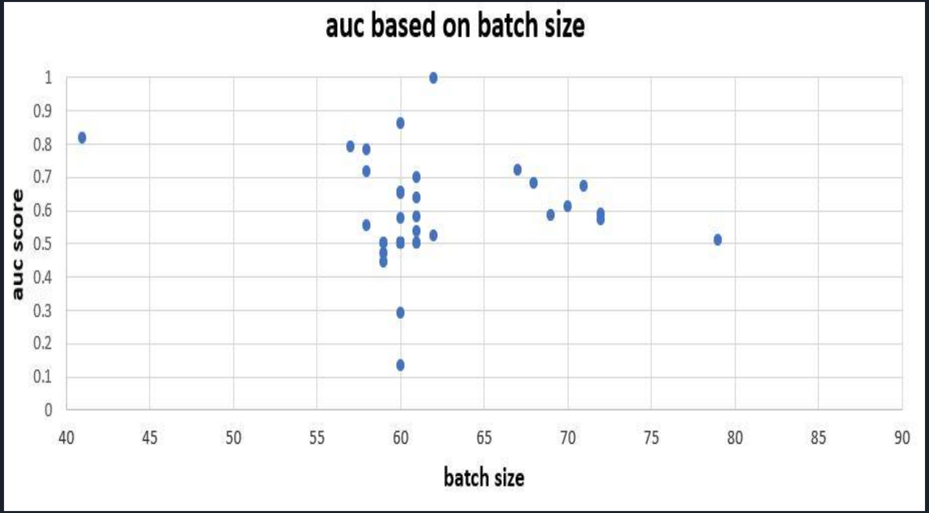
$$f(x) = N(A \times x + b)$$

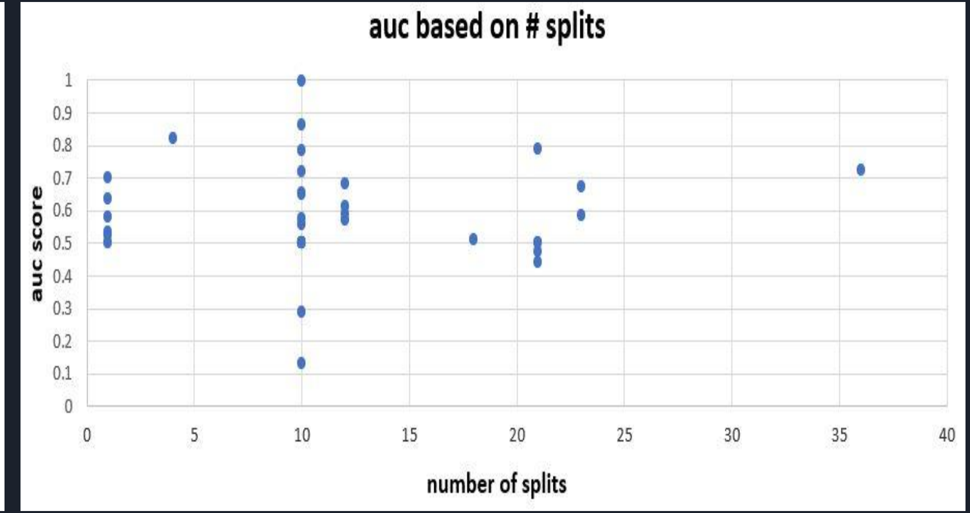
Realisations

gauss, biased, box

Experiment setup and results

- Train data : 40 data sets
- Test data : 10 data sets

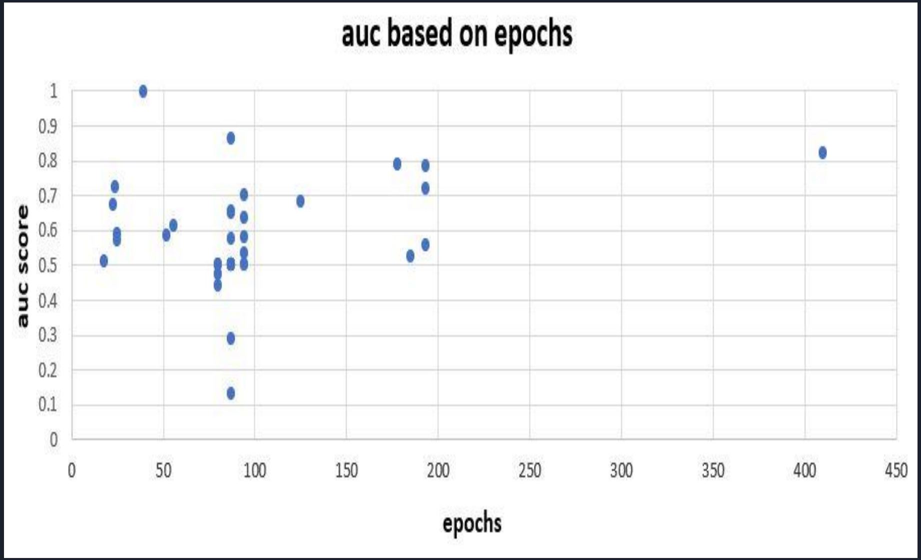




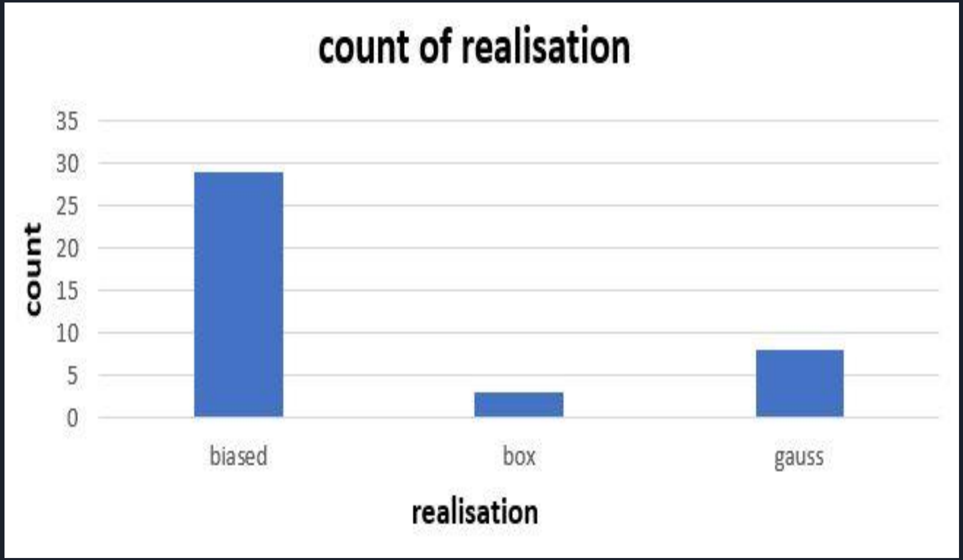
Standard value : 30, Optimized value : 62

Standard value : 10, Optimized value : 11

Experiment setup and results

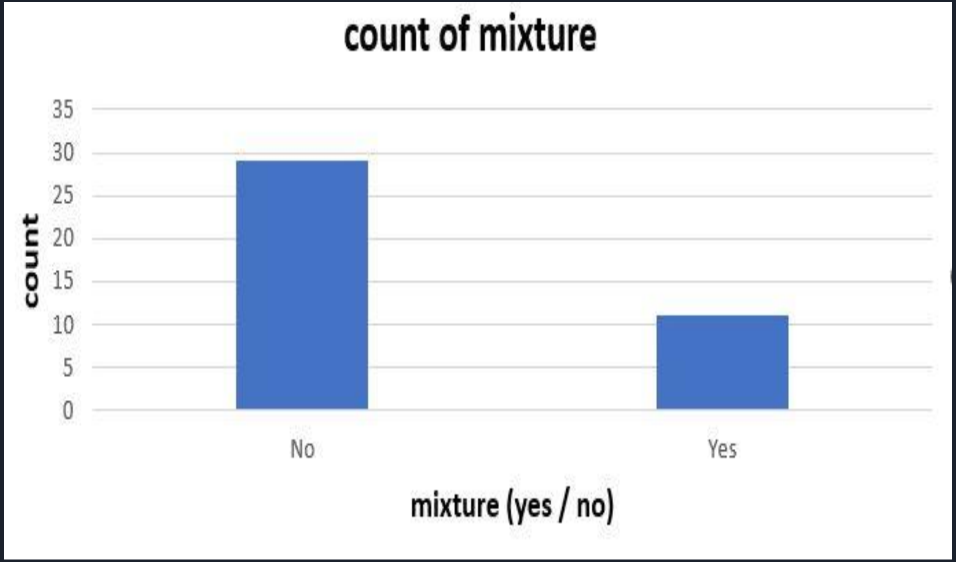


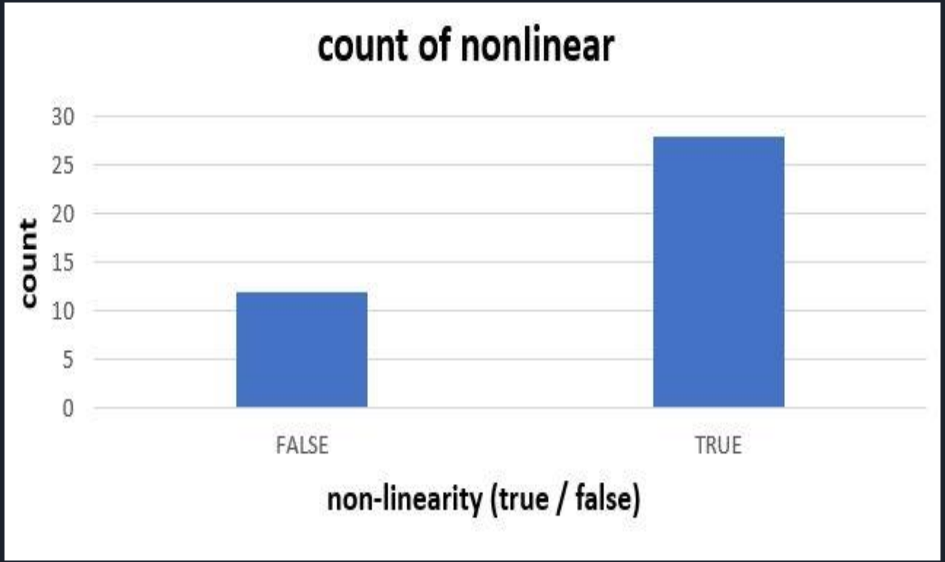
Standard value : 1000, Optimized value : 98



Standard value : “gauss”, Optimized value : “biased”

Experiment setup and results



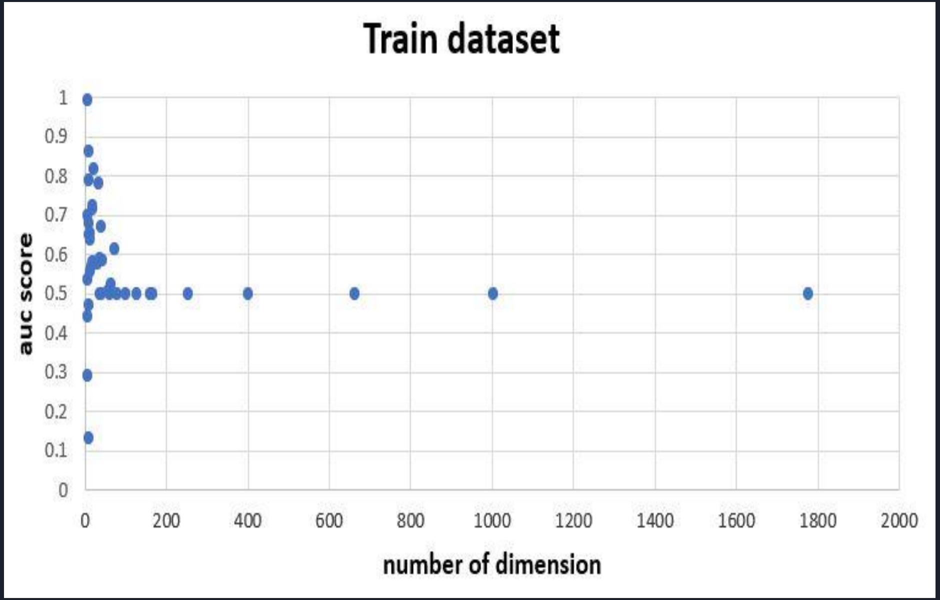


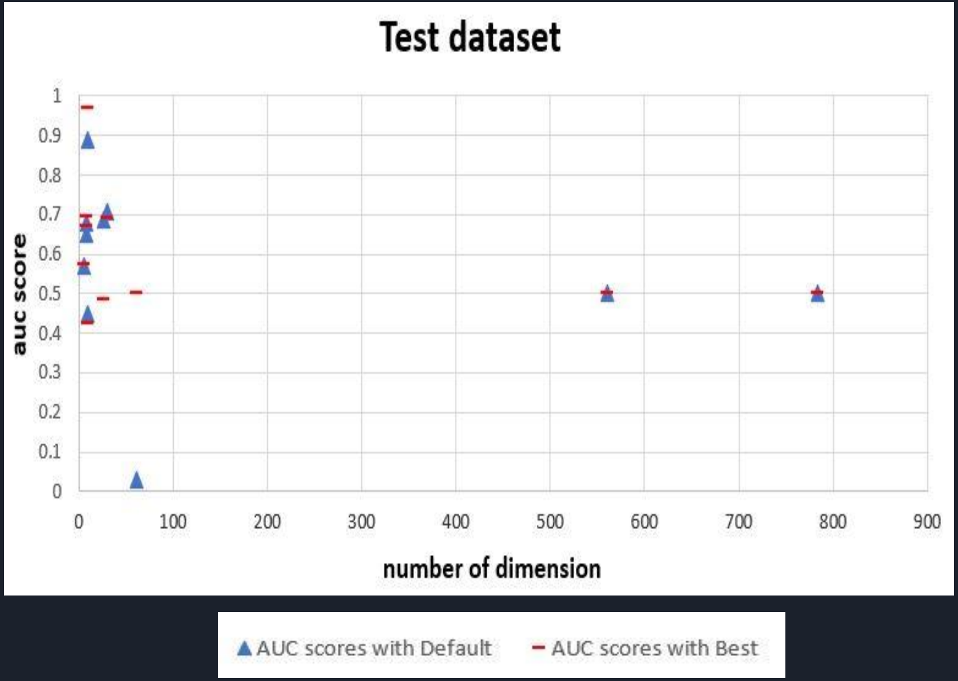
Standard value : No, Optimized value : No

Standard value : False, Optimized value : True

Increased AUC score : 0.600845605 - 0.565866246 = 0.034979358

Interesting behaviour in high dimensions

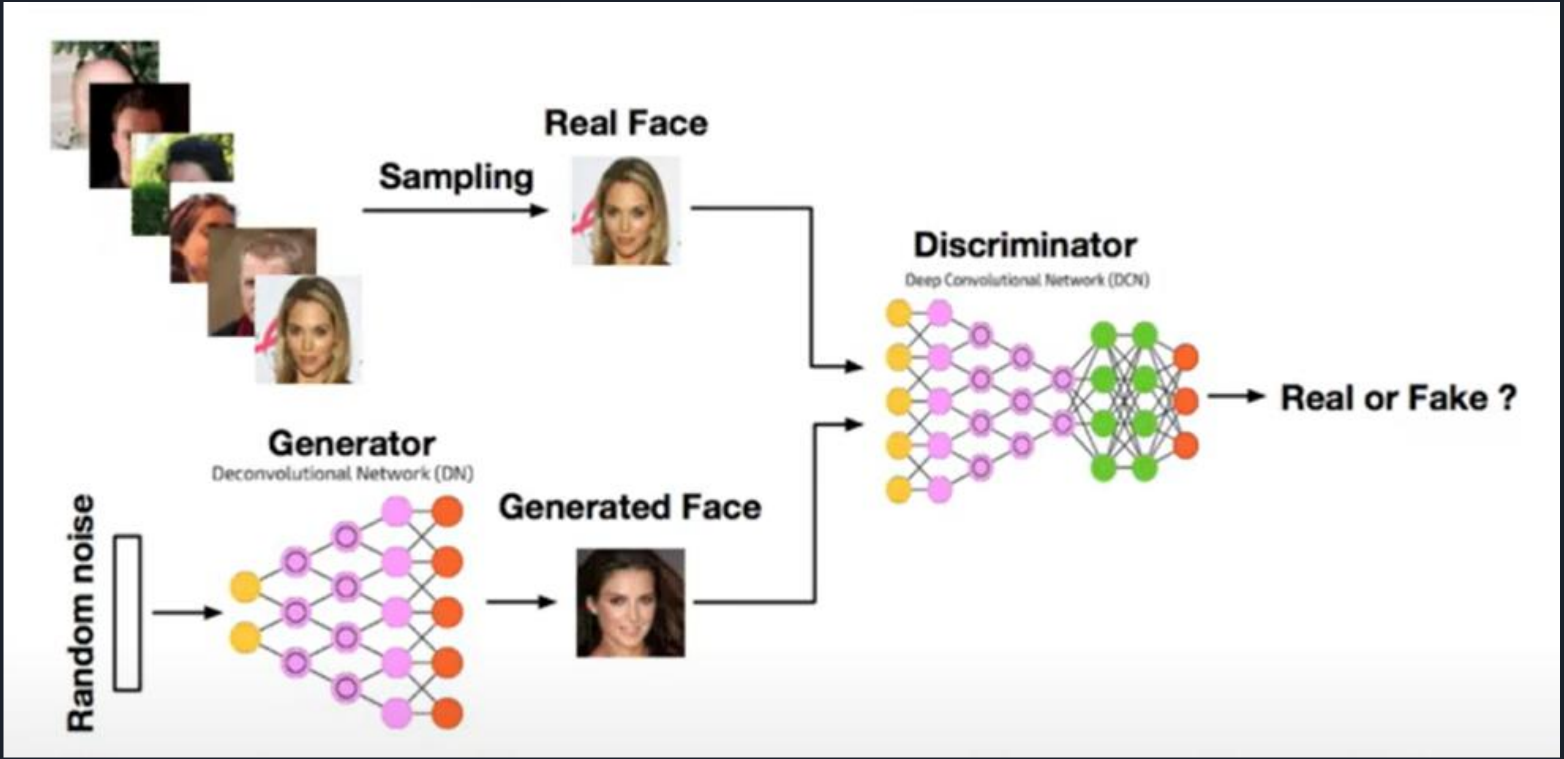




GAN

By Rama Kassoumeh

Generative Adversarial Networks (AnoGAN)



https://www.youtube.com/watch?v=yyqfZfnSzcw

Generative Adversarial Networks (AnoGAN)

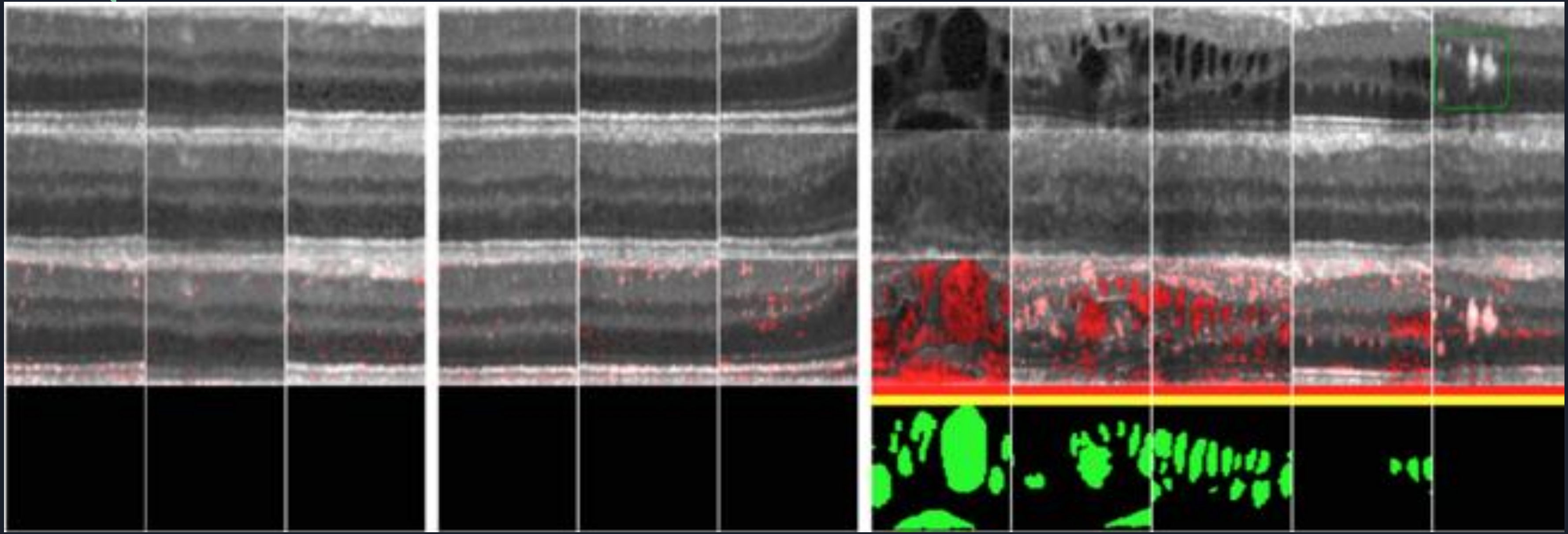
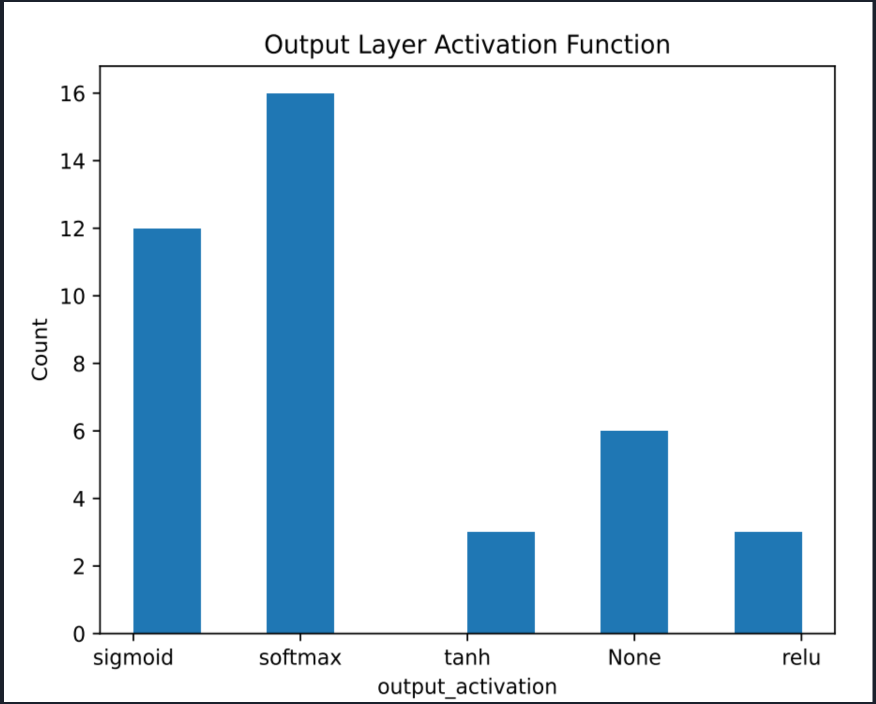
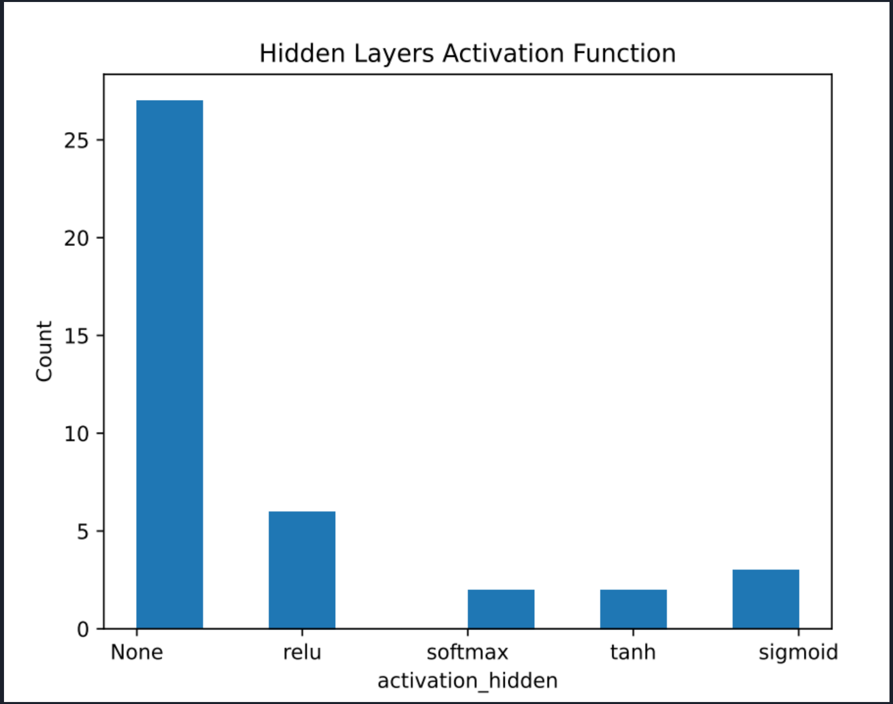


Fig. 3 Unsupervised Anomaly Detection with Generative Adversarial Networks to Guide Marker Discovery

Results (Activation Function)

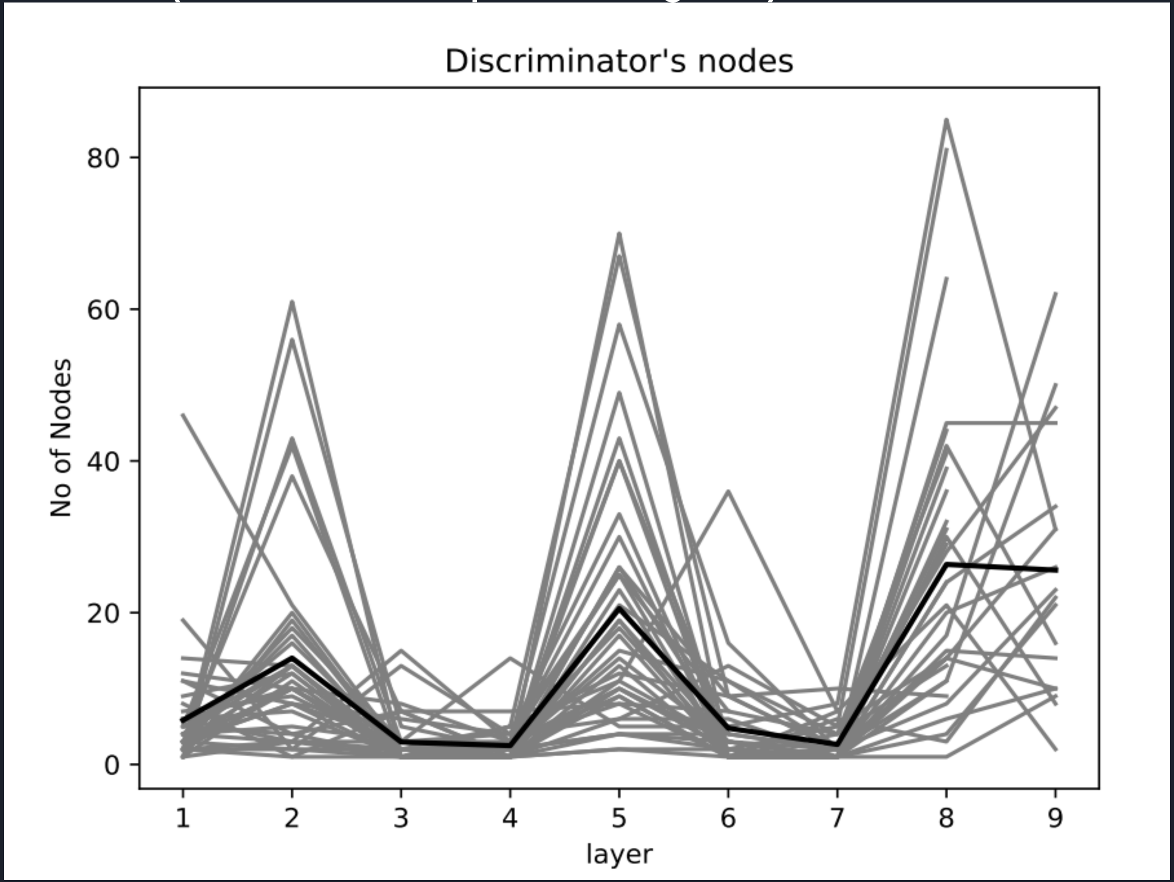


Default = None



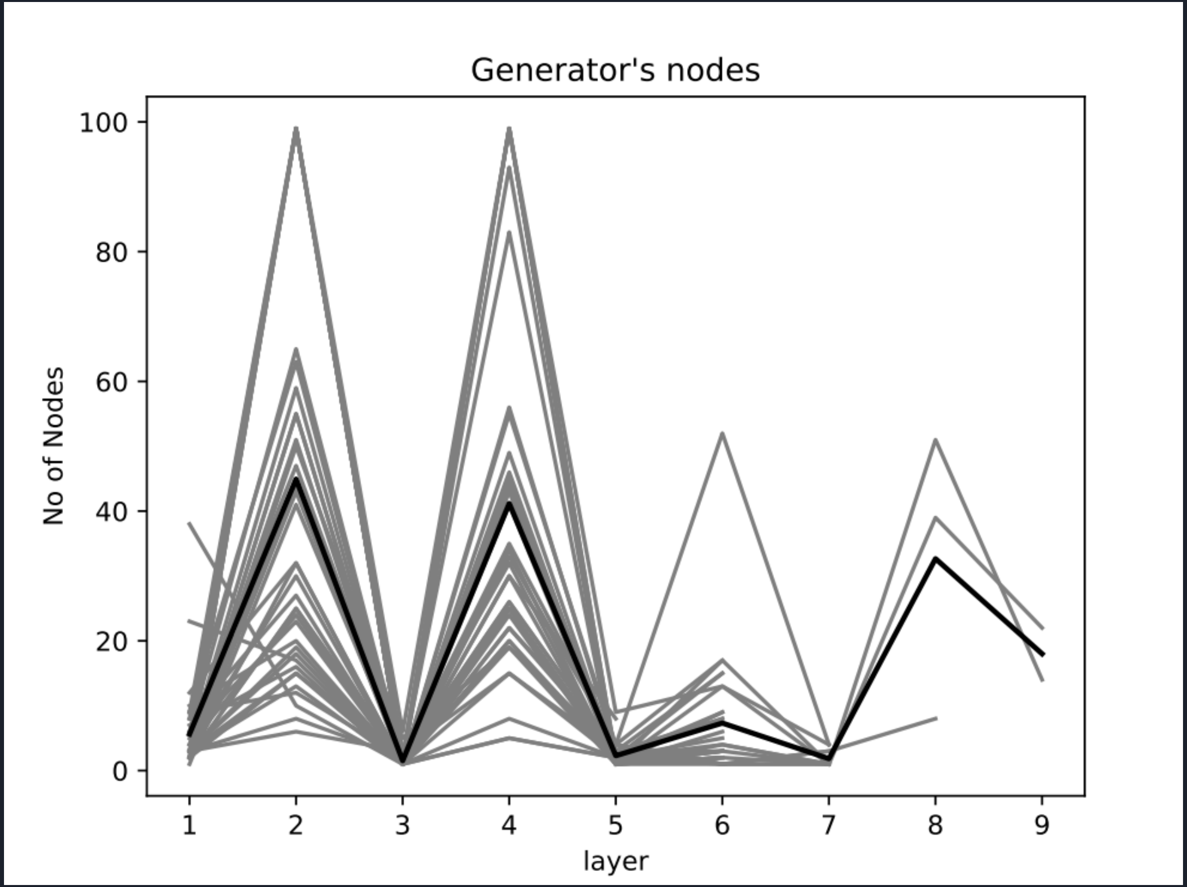
Default = tanh

Results (# Nodes per Layer)



Default = [20, 10, 5]

Results (# Nodes per Layer)

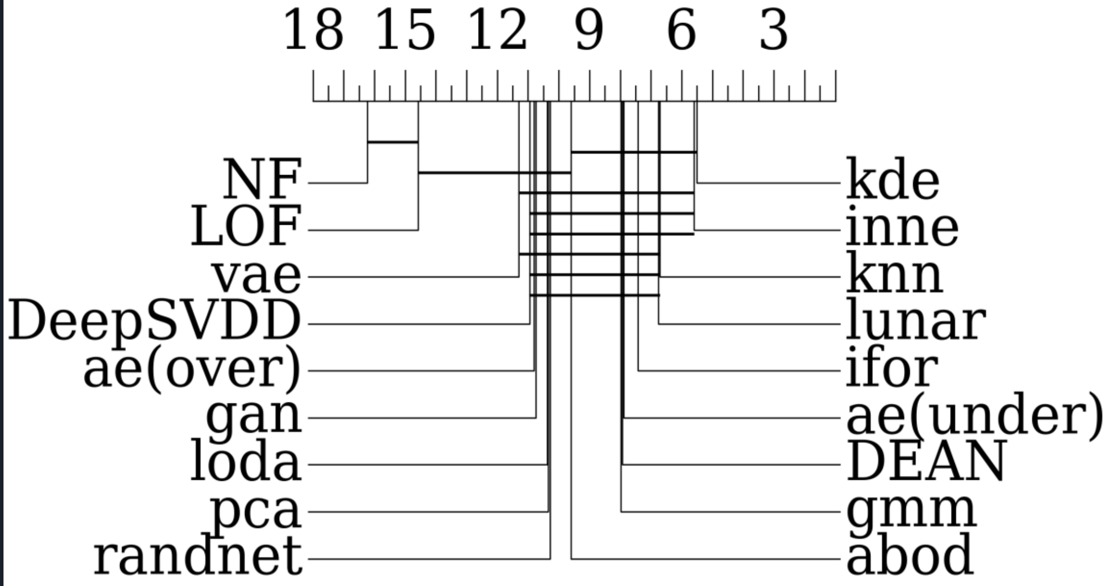


Default = [20, 10, 3, 10, 20]

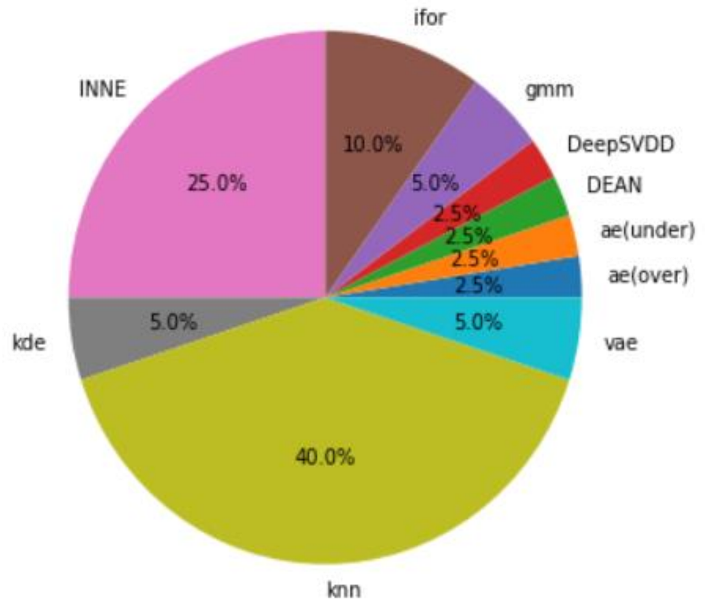
Comparisons

Ranking of Algorithms – Tuned Hyperparameters

Critical Difference Plot

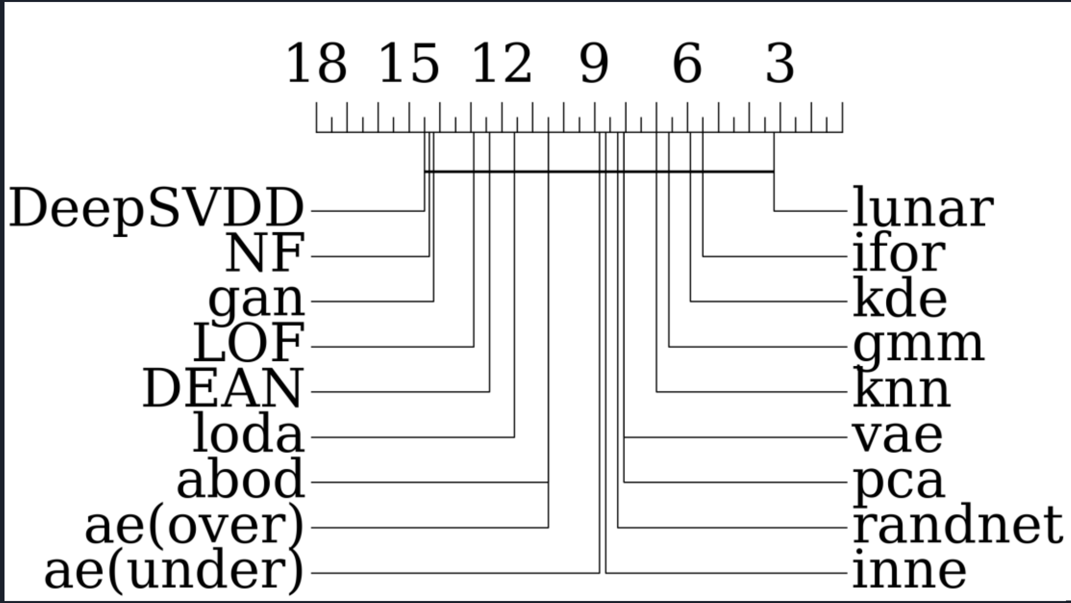


Frequently Best Performing Algorithms

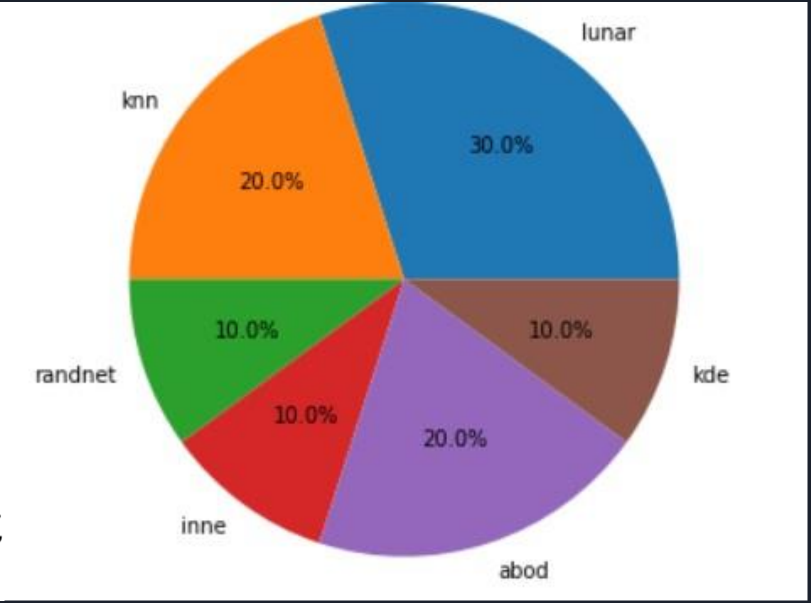


Ranking of Algorithms – Default Hyperparameters

Critical Difference Plot

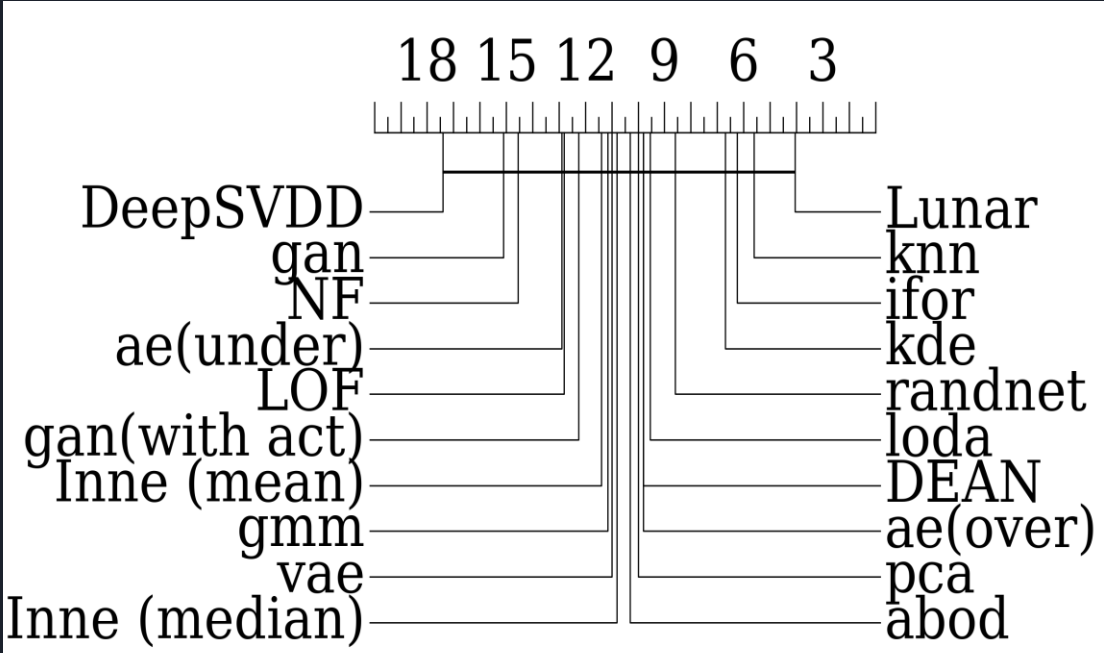


Frequently Best Performing Algorithms

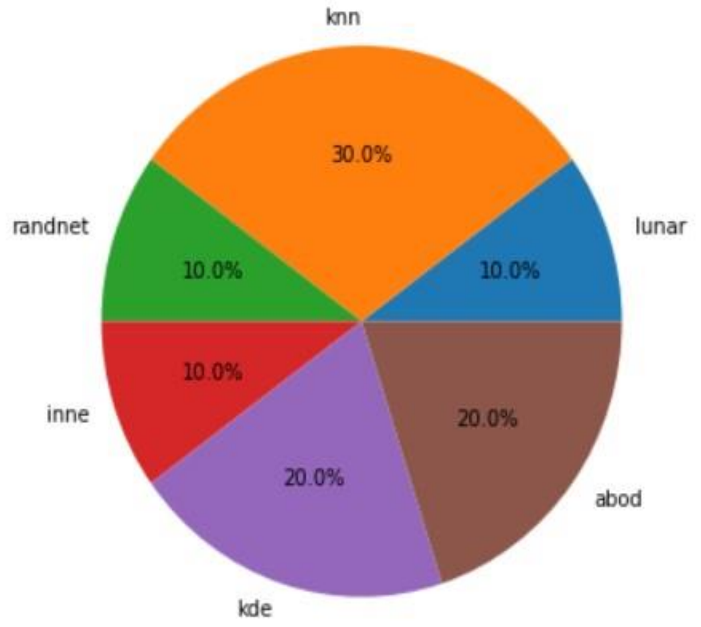


Ranking of Algorithms - Best Hyperparameters

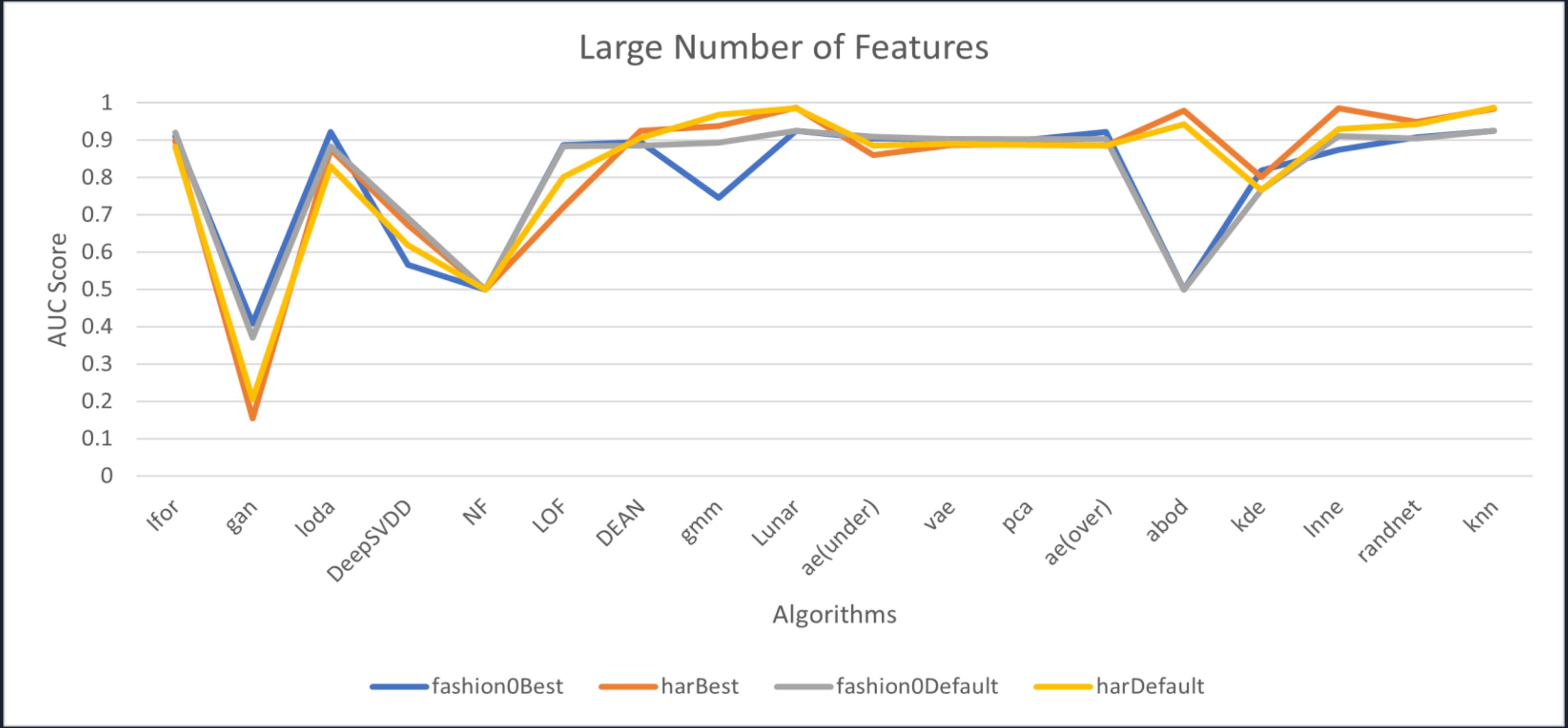
Critical Difference Plot



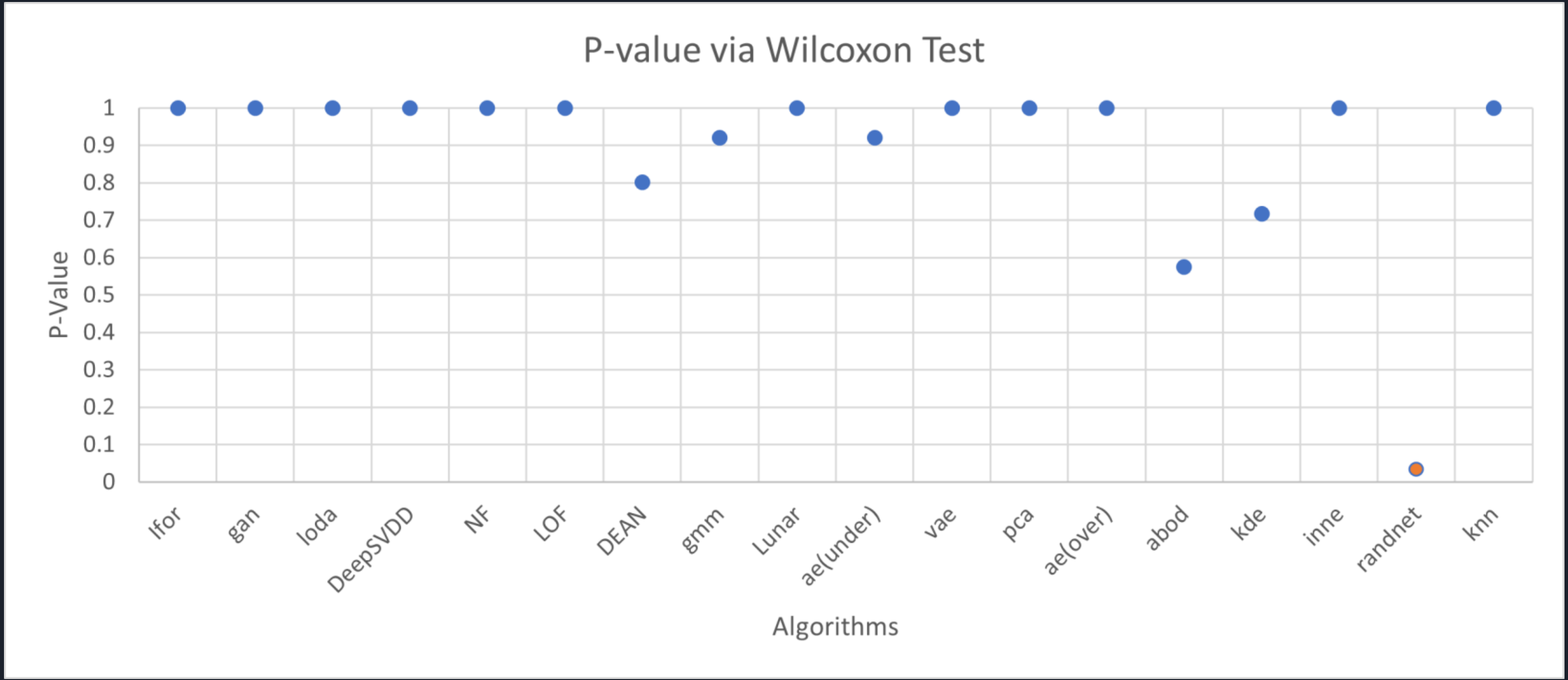
Frequently Best Performing Algorithms



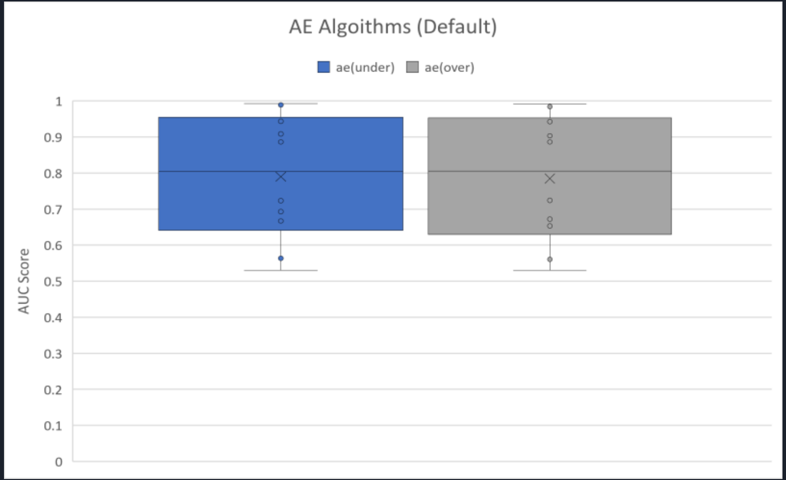
Behavior in higher dimensions

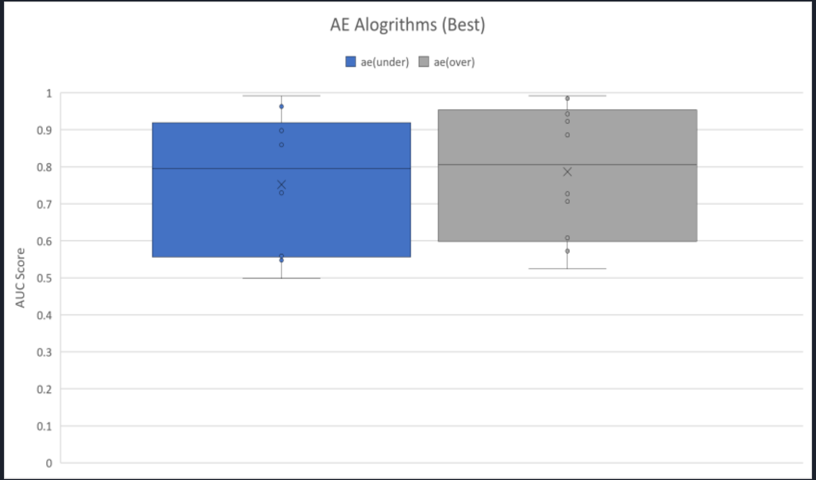


Comparing Best & Default per Algorithms



Comparing AEs





	AE (under) vs AE(over)
Wilcoxon for Default Parameters	0.08
Wilcoxon for Best Parameters	0.04

Summary

Summary

- In GANs, it was observed that a lot of the times the activation function was set to ‘None’. This means that the Neural Network degenerates as there is no activation function involved in any of those times.

- In Normalizing Flows, it was interesting to see that in higher dimensions, the AUC falls down to 0.5. This indicates that the concept of Normalizing Flows does not work in higher dimension datasets.

- In RandNet, as alpha increases the reconstruction power of the algorithm increases as well because increasing alpha leads to increase in model capacity but increasing it too much might cause over fitting. Hence it is beneficial to keep alpha in a moderate range.

Summary

- Understanding the effect of the dataset's shape helps to reduce the number of possible combinations of hyperparameters values.
- Algorithms (Isolation Forest) that randomly sample ***n*** (hyperparameter) features to train on are effected by the number of irrelevant features. Therefore, Understanding the quality of the data helps to determine the right n and reduce the number of possible combinations
- Algorithms that are trained to learn only normal data points (One-class, DEAN) are not able to detect anomalies in the test set if it is too similar to the training set. Therefore, One-class models are not able to detect local anomalies.
- The variance (across the hyperparameter values) of the AUC scores of most Neural Network algorithms is higher than the AUC scores of Nearest Neighbour algorithms. Therefore, It is harder to auto-tune Neural Networks than Nearest Neighbour algorithms

Conclusion

Conclusion

- The drastic changes in results when it comes to the type of pre-processing used is a testament that an autoencoder model is only as good as the data it is being fed
- While number of features could play an important role, it could be argued that it is not the biggest driving aspect when it comes to the accuracy of the autoencoder model
- Focusing on preprocessing techniques based on the distribution of data could lead to better results produced by autoencoders, which ends up making it easier to find anomalies

THANK YOU!