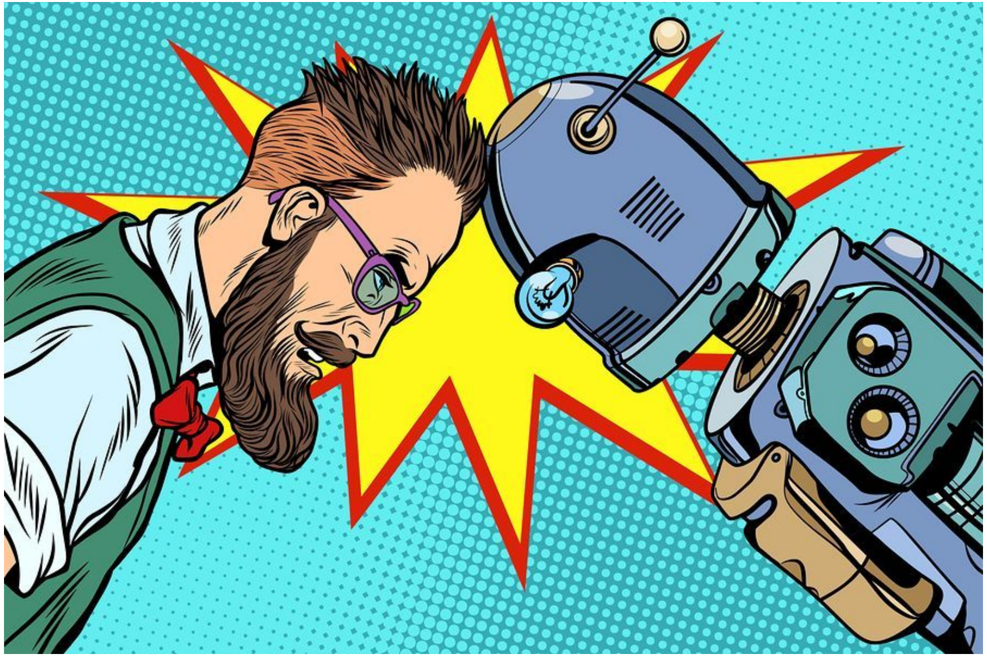
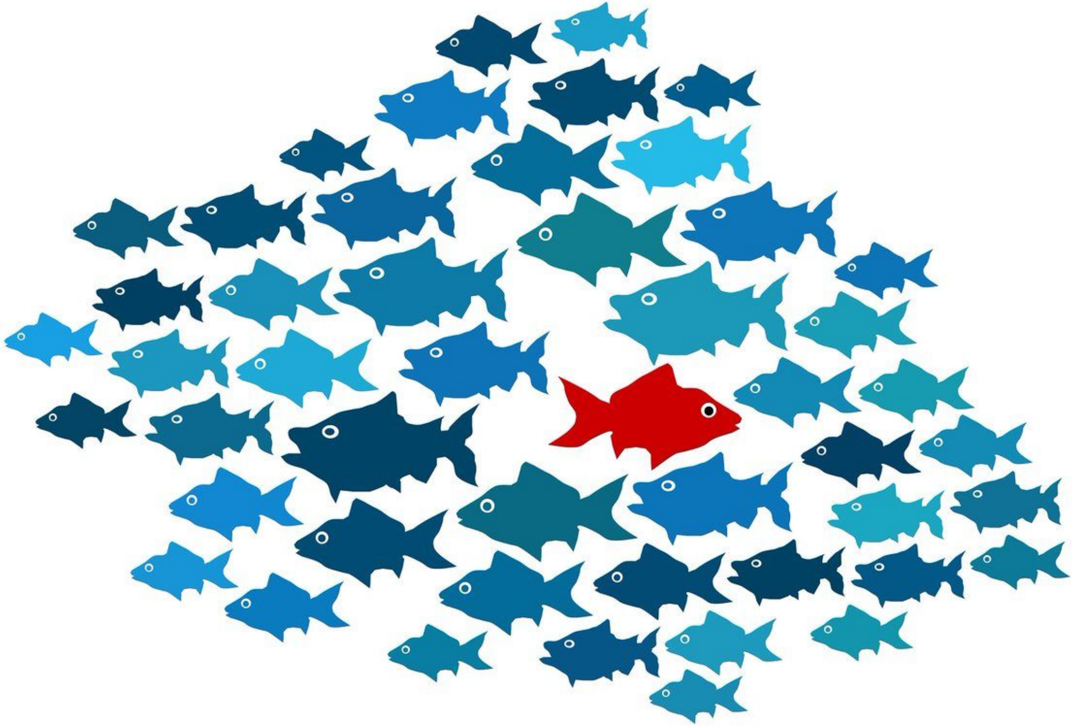
Detección de anomalías



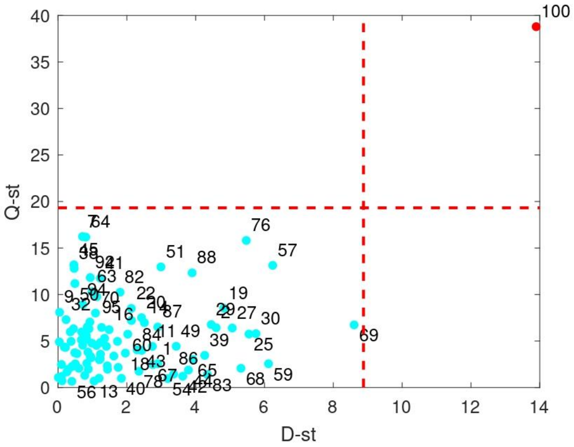
¿Puede la estadística superar a la Inteligencia Artificial?

Anomalías

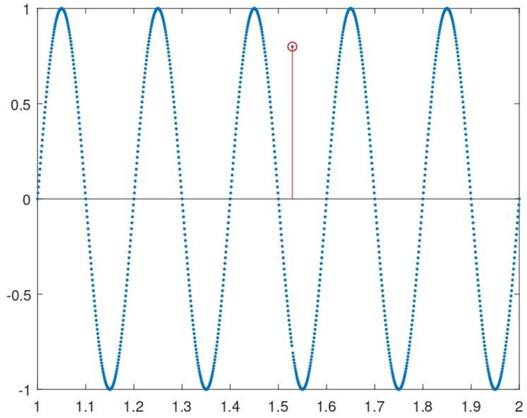


Tipos de anomalías

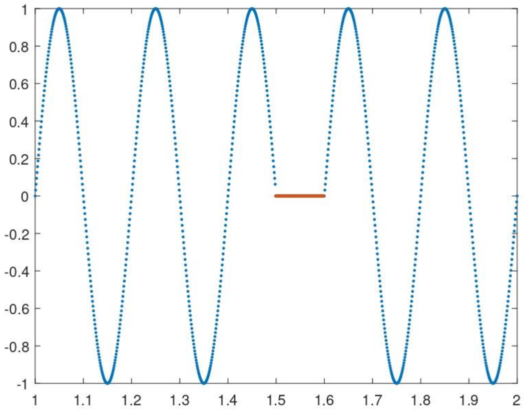
Puntual



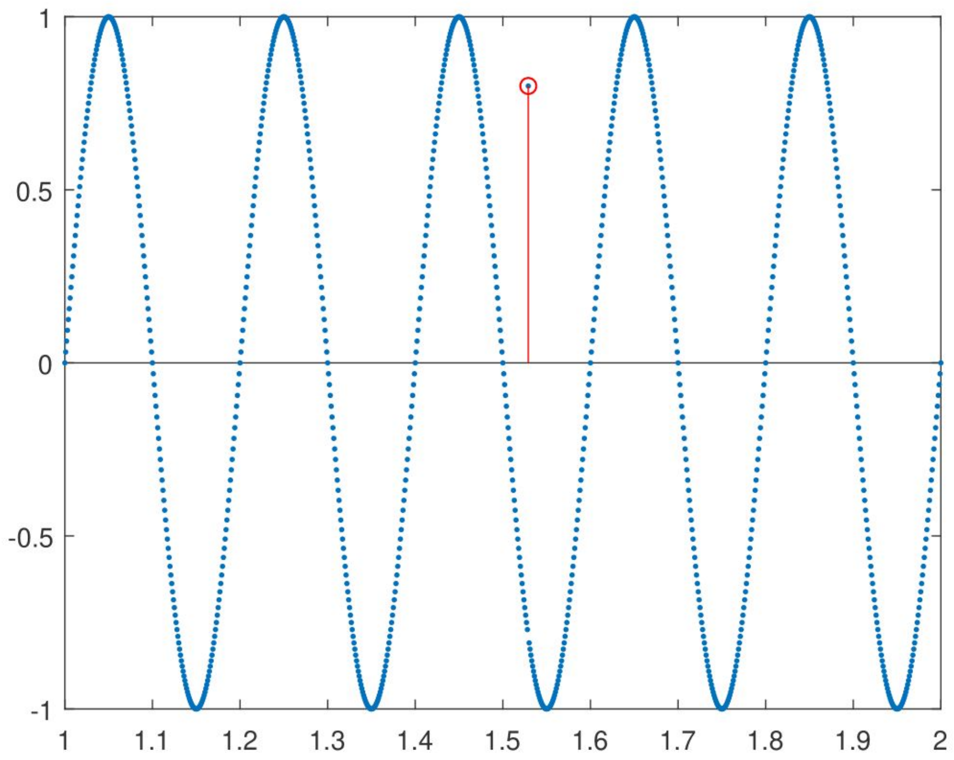
Contextual



Colectivas



¿Qué es la detección de anomalías?

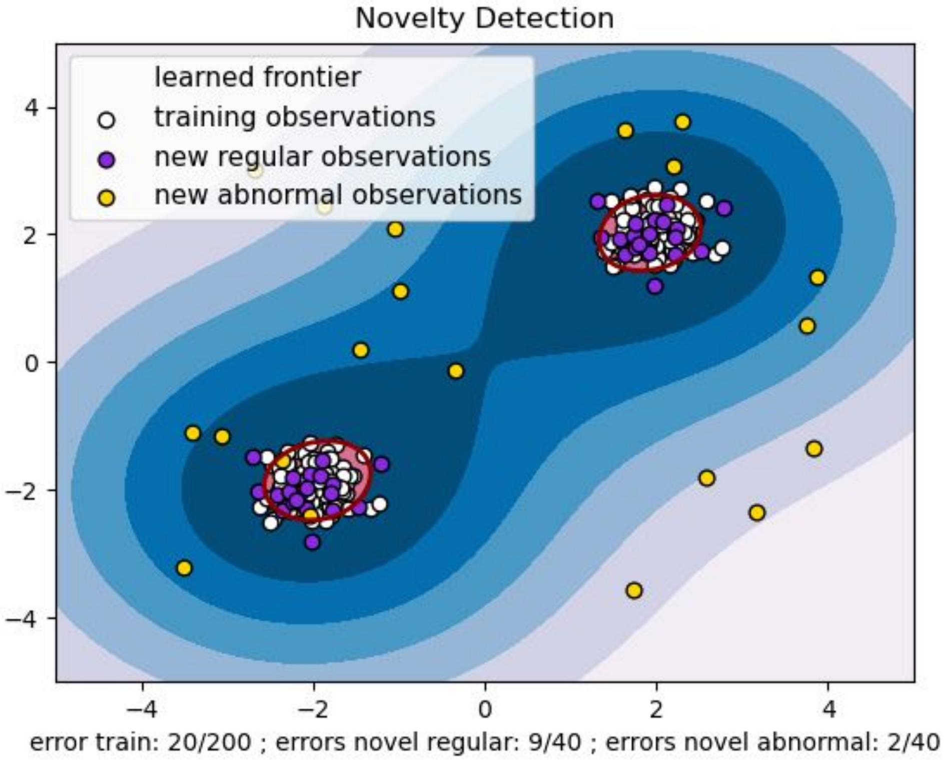


Dataset Multivariante

Tabla 4.1: Dataset reducido países europeos COVID-19.

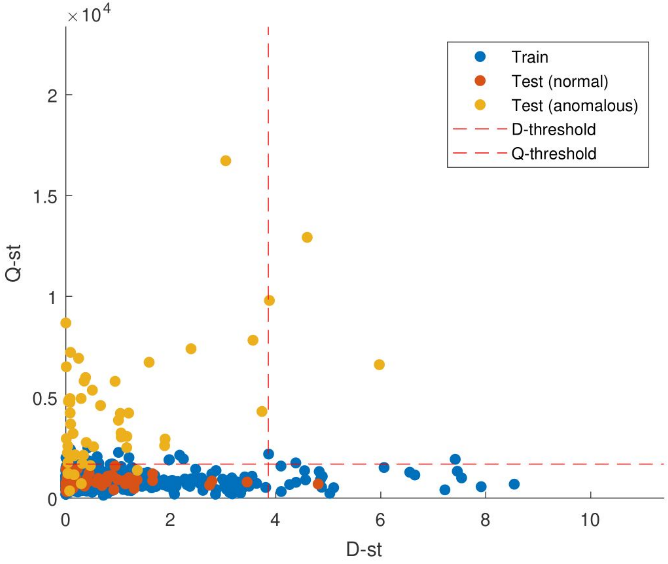
Country	Median age	Cases per million	Deaths per million
DEU	46,6	306698,552	1638,835
FRA	42,0	433655,706	2185,399
GBR	40,8	326289,747	2601,2680
ITA	47,9	282567,295	2737,302
ESP	45,5	259430,255	2255,718
POL	41,8	158833,664	3074,503
NLD	43,2	475238,184	1304,308
CHE	43,1	418697,437	1581,207
FIN	42,8	192802,884	772,120
NOR	39,7	261593,130	560,045
LTU	43,5	394302,384	3393,111
SEN	18,7	5003,341	114,327

Modelo de Normalidad



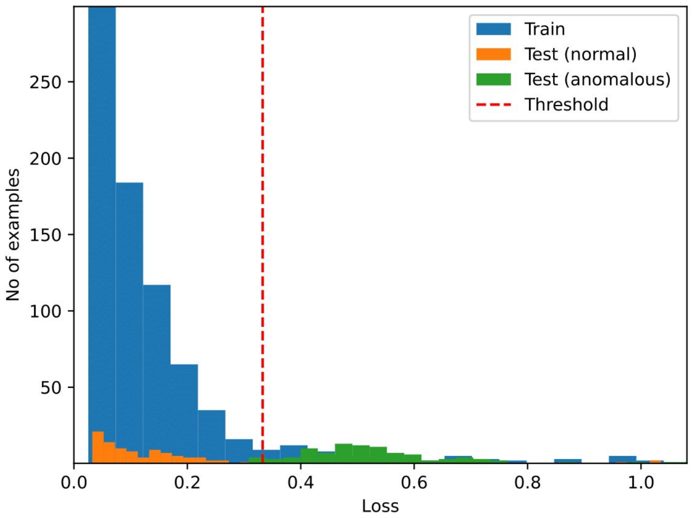
Objetivo

Multivariate analysis

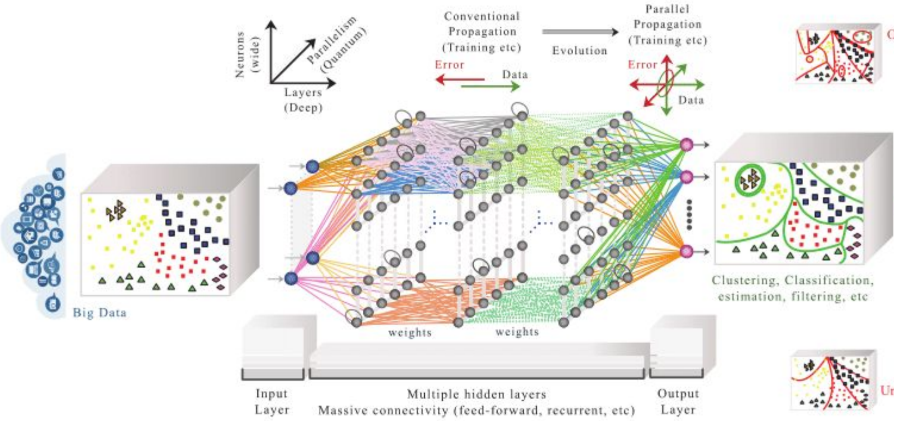


VS

Deep learning



Anécdota

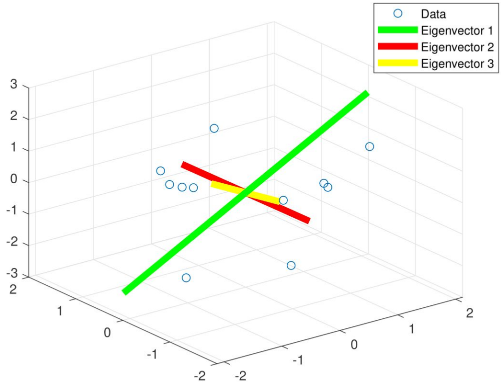



```sql
SELECT menu_name, COUNT(*) as menu_count FROM usage_logs
GROUP BY menu_name ORDER BY menu_count DESC LIMIT 5;
```

Métodos

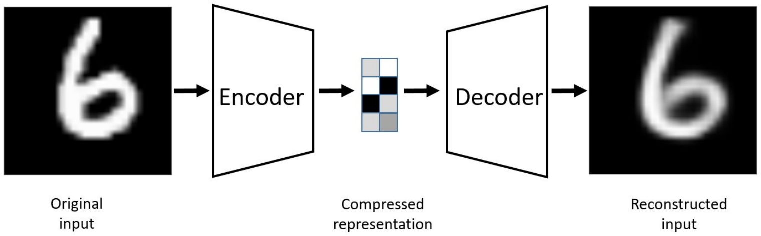
Multivariate análisis

PCA



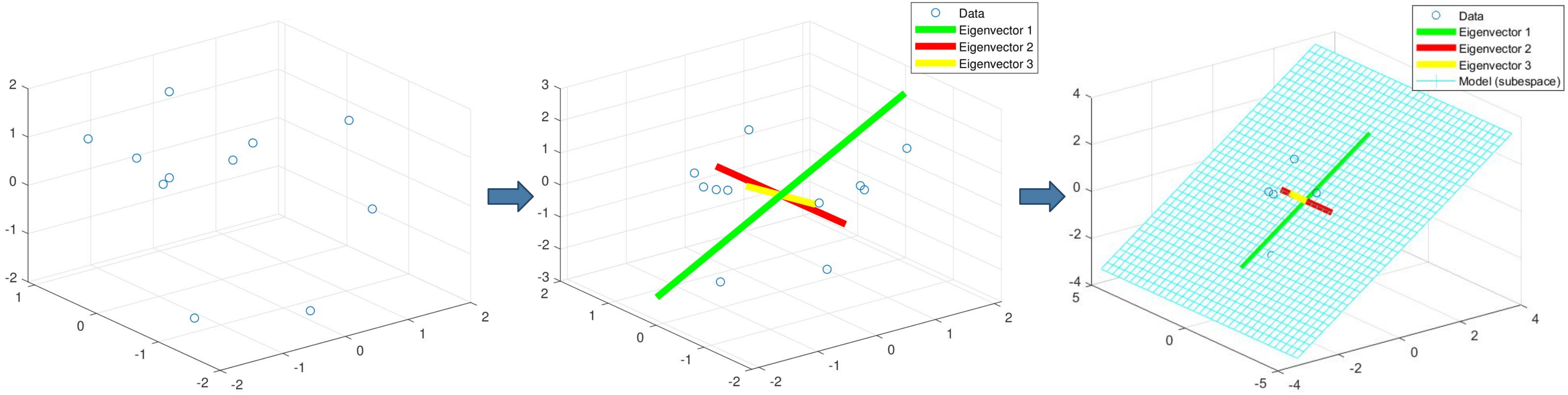
Deep learning

Autoencoders



[D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” 2020]

PCA



$$X \rightarrow P \rightarrow T \rightarrow P^T \rightarrow \hat{X}$$

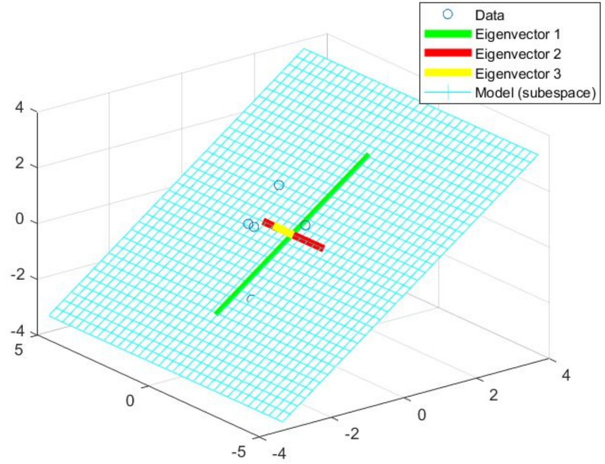
Data in · **Loadings** · **Scores** · **Loadings'** · **Reconstructed data**

Entrenamiento de PCA

Algoritmo 1 Proceso de entrenamiento de PCA.

Require: *train_data* to be normal
Ensure: Minimal loss

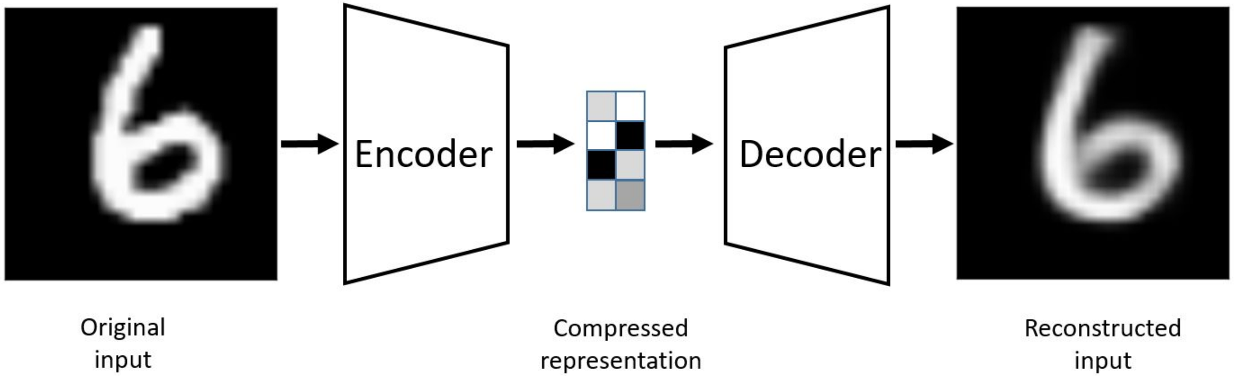
```
procedure TRAIN(train_data, PCs)
    X ← train_data                        ▷ Without labels
    P ← eigenvectors(X · X^T)             ▷ Loadings
    D ← eigenvalues(X · X^T)
    Sort P by D
    T ← X · P                             ▷ Scores
    D ← D(1 : PCs)
    T ← T(1 : PCs)                        ▷ Use only first PCs
end procedure
```



$$X \rightarrow P \rightarrow T \rightarrow P^T \rightarrow \hat{X}$$

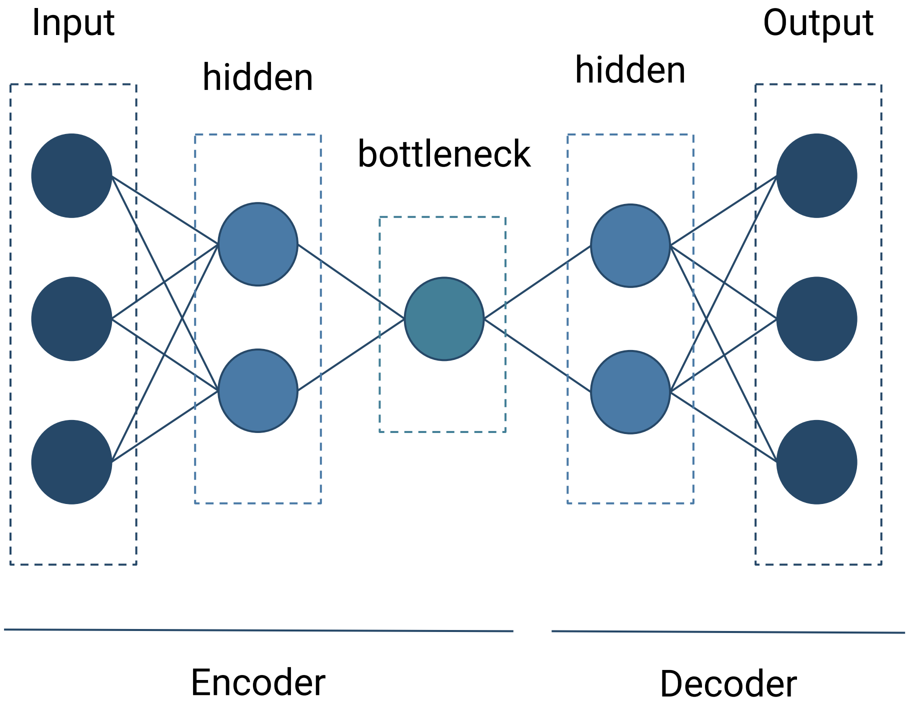
Data in — Loadings — Scores — Loadings' — Reconstructed data

Autoencoder



[D. Bank, N. Koenigstein, and R. Giryes, “Autoencoders,” 2020]

Autoencoder



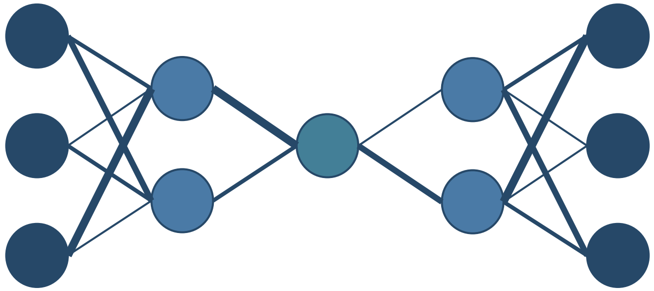
MSE(Output, Input) ≈ 0

Entrenamiento del Autoencoder

Algoritmo 2 Proceso de entrenamiento del Autoencoder.

Require: *train_data* to be normal
Ensure: Minimal loss

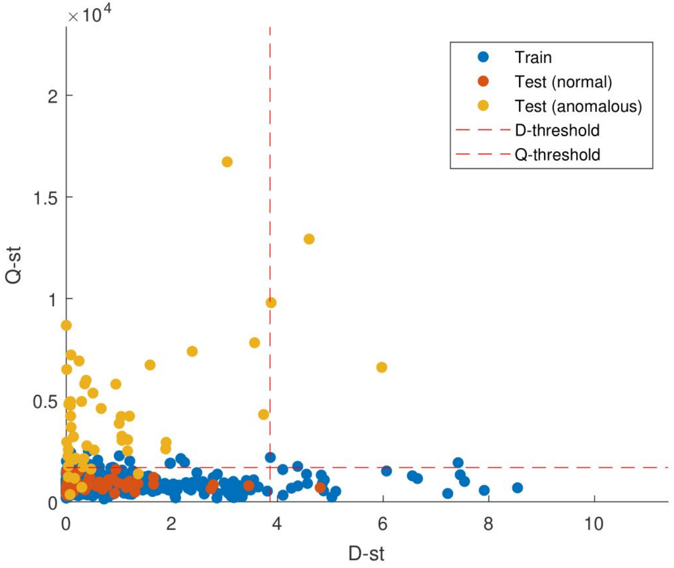
procedure TRAIN(train_data, epochs, bottleneck_size, hidden_size)
    $X \leftarrow train\_data$ ▷ Without labels
    $N \leftarrow epochs$
    Build *encoder* network with provided sizes
    Build *decoder* network with provided sizes
    for N iterations **do**
        $X_{enc} \leftarrow Encoder(X)$
        $X_{dec} \leftarrow Decoder(X_{enc})$
        $loss \leftarrow MSE(X, X_{dec})$
        $Encoder, Decoder \leftarrow ADAM(Encoder, Decoder, loss)$
    end for
    $Model \leftarrow Decoder(Encoder(X))$
end procedure



$$X \rightarrow Encode \rightarrow Bottleneck \rightarrow Decode \rightarrow \hat{X}$$

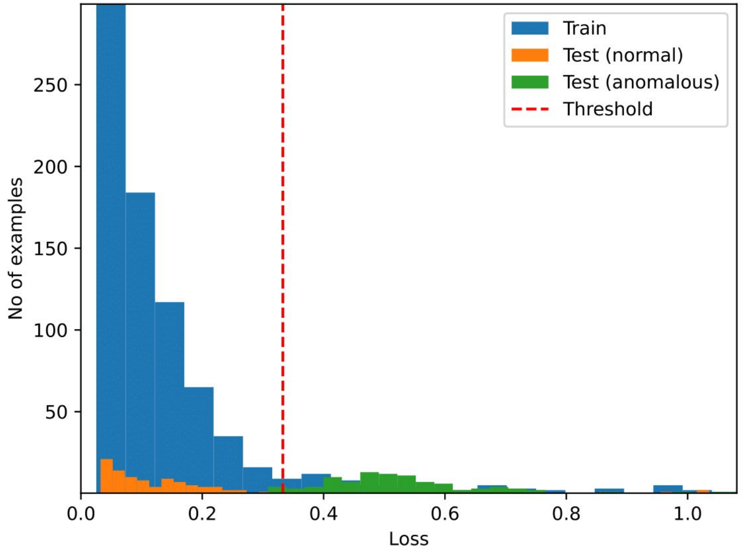
Detección

PCA

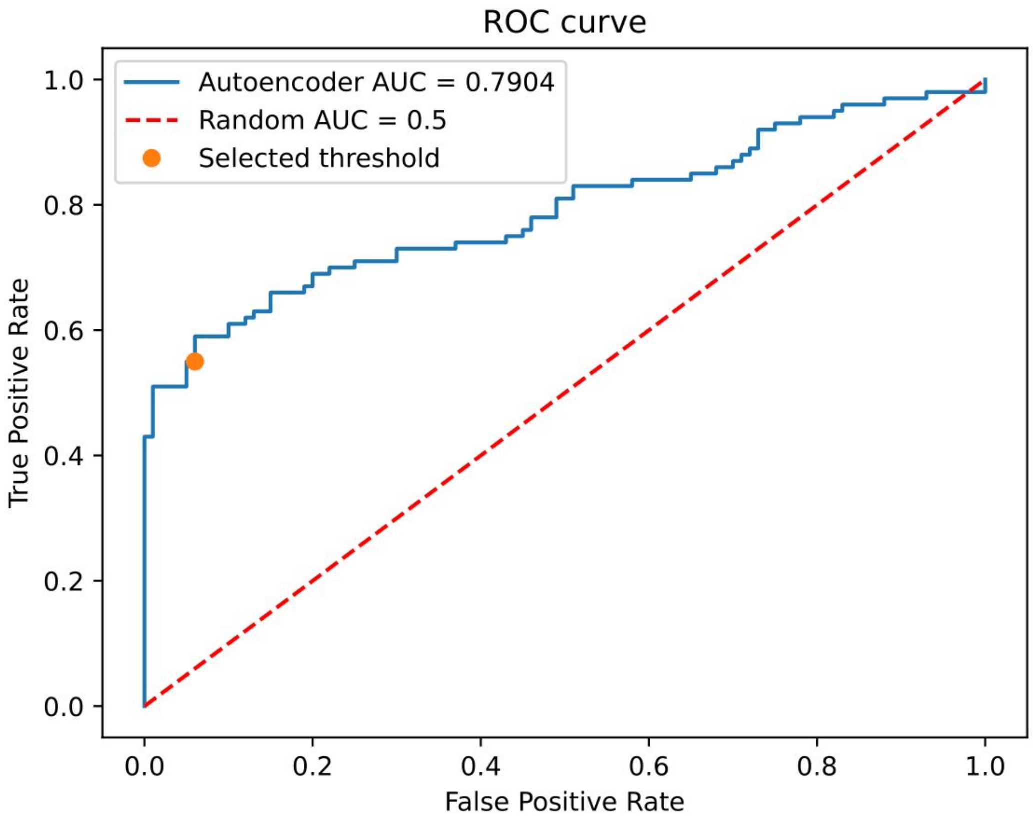


$$T(n) = \alpha \frac{D(n)}{UCL_P^D} + (1+\alpha) \frac{Q(n)}{UCL_P^Q}$$

Autoencoder

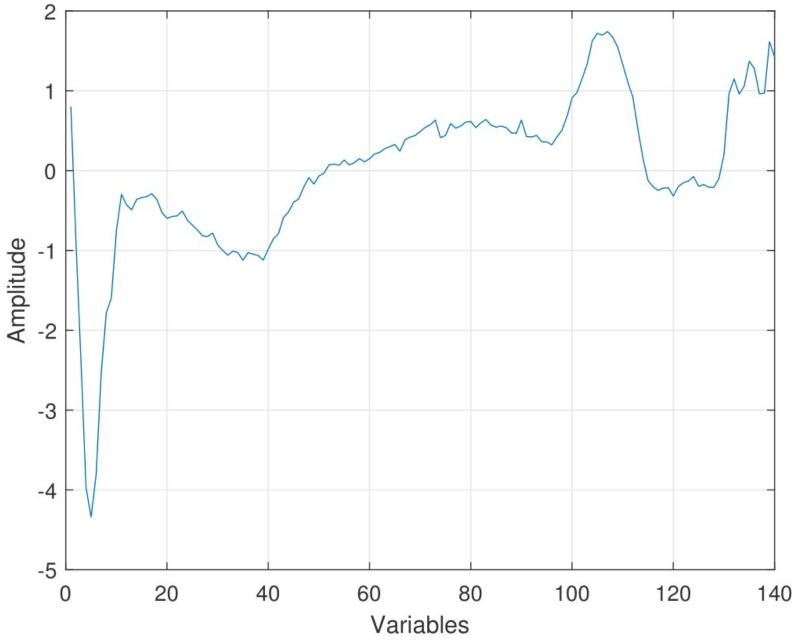


Métricas: ROC y AUC

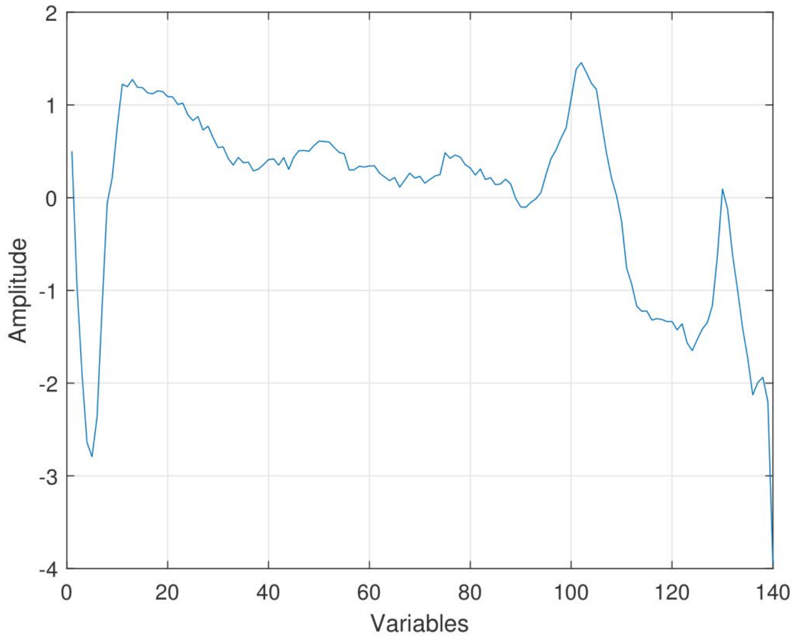


Dataset ECG

Normal



Anomalía



Resultados ECG

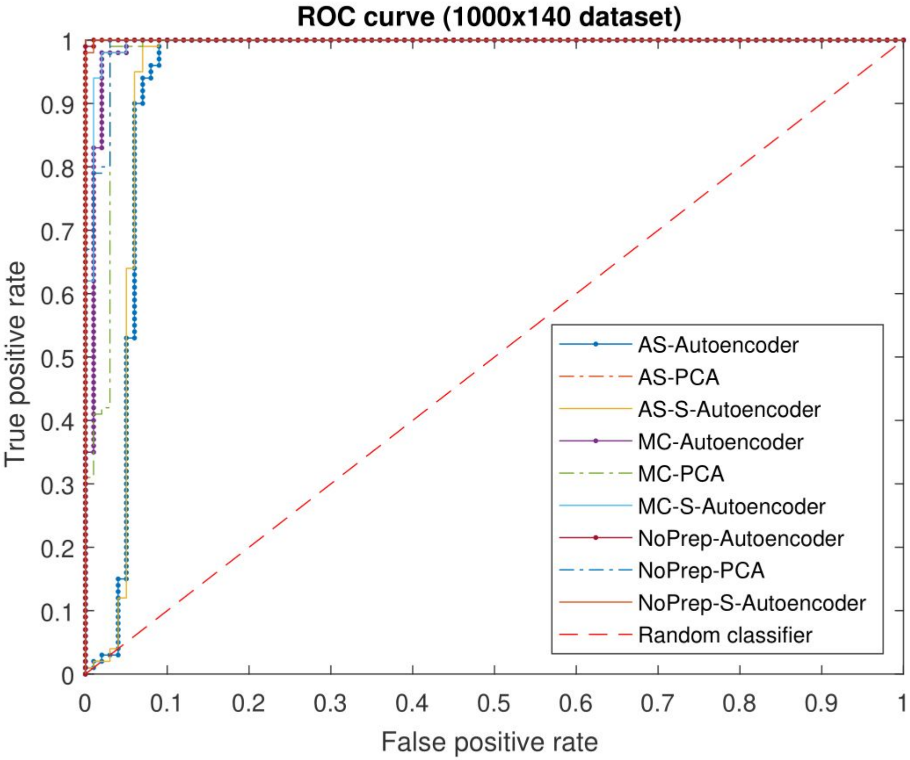


Tabla 6.4: Resultados del *dataset* de electrocardiogramas con distintos pre-procesados: *Auto-scaling* (AS), *Mean-centering* (MC) y sin procesar (∅).

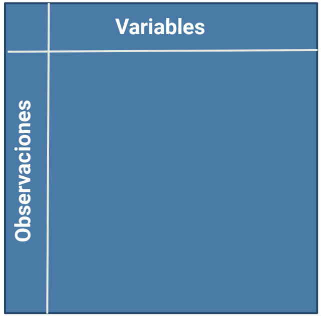
Prep.	**Method**	**AUC**	**Precision**	**Recall**	**Accuracy**	F_1
AS	PCA	**1**	0.89	1	0.945	0.9418
	Autoencoder	0.9453	0.99	0.93396	0.96	0.96117
	S-Autoencoder	0.9485	0.98	0.93333	0.955	0.9561
MC	PCA	0.981	0.99	0.97059	0.98	0.9802
	Autoencoder	0.9912	0.74	0.98667	0.865	0.84571
	S-Autoencoder	**0.995**	0.81	0.9878	0.9	0.89011
∅	PCA	0.9926	0.99	0.97059	0.98	0.9802
	Autoencoder	**0.9999**	0.99	0.99	0.99	0.99
	S-Autoencoder	0.9998	0.98	0.9899	0.985	0.98492

Generación de *datasets*

Delgado

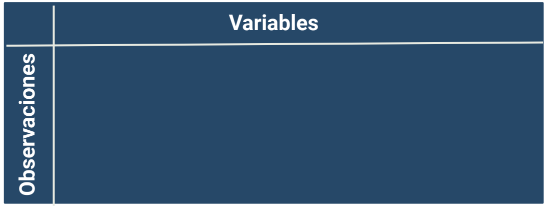
1000 x 100

Cuadrado



1000 x 1000

Grueso



1000 x 10000

Dataset delgado

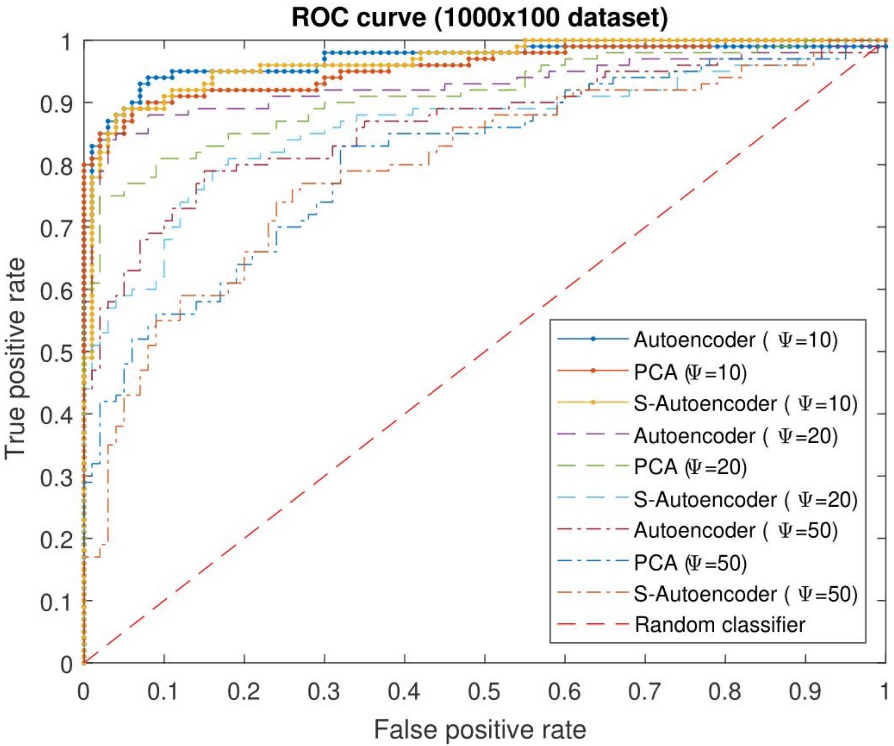


Tabla 6.1: Resultados del *dataset* delgado (100 variables) con Ψ variables anómalas.

Ψ	**Method**	**AUC**	**Precision**	**Recall**	**Accuracy**	**F_1**
10	PCA	0.9569	0.82	0.97619	0.9	0.8913
	Autoencoder	**0.9678**	0.93	0.92079	0.925	0.92537
	S-Autoencoder	0.9645	0.94	0.85455	0.89	0.89524
20	PCA	0.9126	0.62	0.96875	0.8	0.7561
	Autoencoder	**0.9272**	0.87	0.91579	0.895	0.89231
	S-Autoencoder	0.8608	0.82	0.78846	0.8	0.80392
50	PCA	0.8101	0.34	0.94444	0.66	0.5
	Autoencoder	**0.864**	0.68	0.90667	0.805	0.77714
	S-Autoencoder	0.7968	0.61	0.75309	0.705	0.67403

Dataset cuadrado

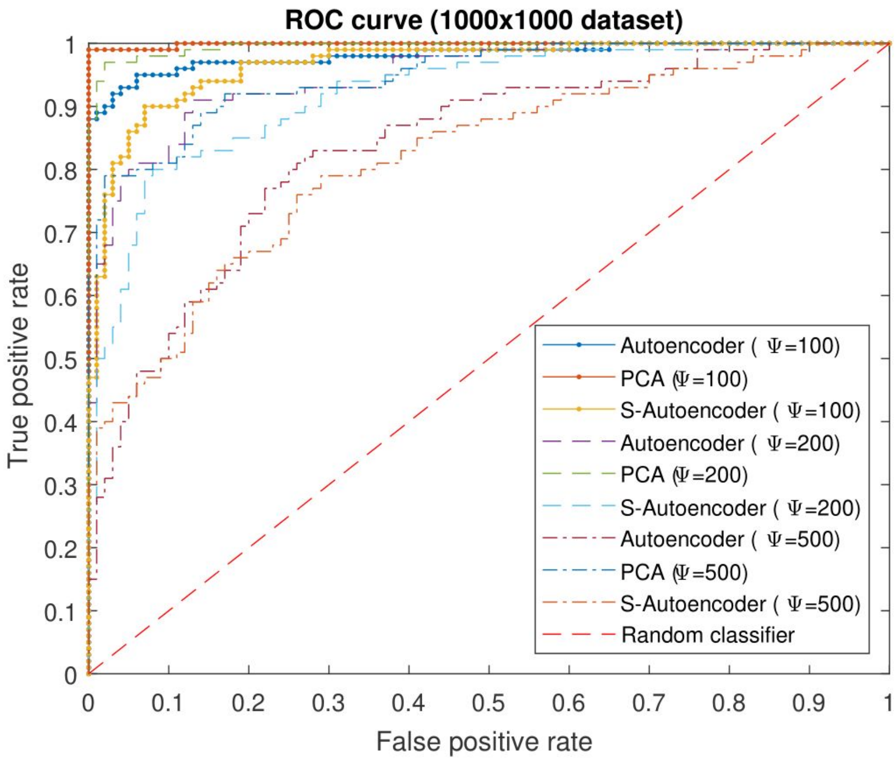


Tabla 6.2: Resultados del *dataset* cuadrado (1000 variables) con Ψ variables anómalas.

Ψ	Method	AUC	Precision	Recall	Accuracy	F_1
	PCA	**0.9989**	0.99	0.99	0.99	0.99
100	Autoencoder	0.9815	0.99	0.66892	0.75	0.79839
	S-Autoencoder	0.9667	0.97	0.78226	0.85	0.86607
	PCA	**0.9953**	0.92	0.98925	0.955	0.95337
200	Autoencoder	0.9464	0.99	0.66443	0.745	0.79518
	S-Autoencoder	0.9201	0.92	0.74797	0.805	0.82511
	PCA	**0.9483**	0.68	0.98551	0.835	0.80473
500	Autoencoder	0.8355	0.93	0.63699	0.7	0.7561
	S-Autoencoder	0.8159	0.85	0.66406	0.71	0.74561

Dataset grueso

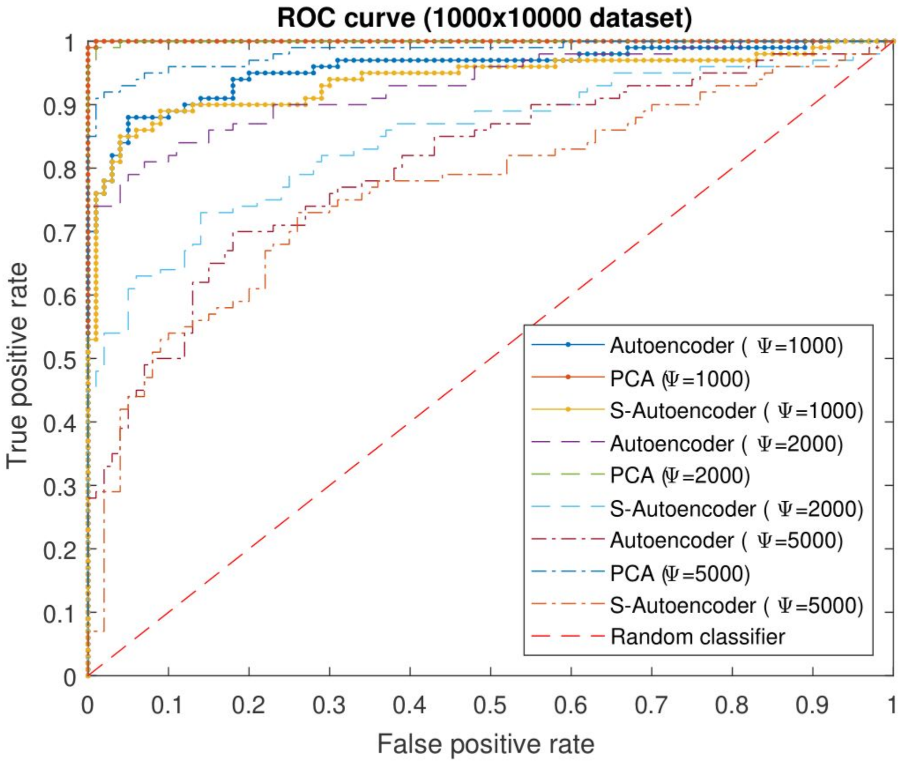


Tabla 6.3: Resultados del *dataset* grueso (10000 variables) con Ψ variables anómalas.

Ψ	Method	AUC	Precision	Recall	Accuracy	F_1
1000	PCA	**0.9999**	1	0.87719	0.93	0.93458
	Autoencoder	0.9564	0.81	0.96429	0.89	0.88043
	S-Autoencoder	0.9385	0.85	0.95506	0.905	0.89947
2000	PCA	**0.9993**	1	0.87719	0.93	0.93458
	Autoencoder	0.93	0.74	0.94872	0.85	0.83146
	S-Autoencoder	0.8481	0.57	0.91935	0.76	0.7037
5000	PCA	**0.9838**	0.96	0.87273	0.91	0.91429
	Autoencoder	0.805	0.37	0.90244	0.665	0.52482
	S-Autoencoder	0.7664	0.47	0.85455	0.695	0.60645

Conclusiones

Flexibilidad

Autoencoder se adapta mejor a *datasets* no lineales

Escalabilidad

PCA mayor consistencia con el aumento de variables

Coste

PCA menor coste de entrenamiento e inferencia

Referencias

- J. Camacho, J. M. García-Giménez, N. M. Fuentes-García, and G. Maciá-Fernández, "Multivariate big data analysis for intrusion detection: 5 steps from the haystack to the needle" 2019
- D. Bank, N. Koenigstein, and R. Giryes, "Autoencoders" 2020
- J. Camacho. MEDA-Toolbox (https://github.com/josecamachop/MEDA-Toolbox), 2022
- M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems" 2015

Agradecimientos

- José Camacho Páez (Universidad de Granada)
- José Suárez-Varela Maciá (Universidad Politécnica de Cataluña)