

DEEP LEARNING: AUTOENCODERS FOR ANOMALY DETECTION № 9

Daniela Pinto Veizaga, dpintove@itam.mx Diego Villa Lizárraga, dvillali@itam.mx

Introducción

En objetivo del presente trabajo es:

- entrenar un modelo de aprendizaje no supervisado mediante el uso de autoencoders (AE);
- usar un AE para estimar representaciones latentes de algunos datos numéricos;
- utilizar esas representaciones latentes para detectar anomalías.

Para cumplir con los objetivos anteriores, en el presente trabajo intentamos detectar anomalías o transacciones fraudulentas con el data set "Credit Card Fraud", disponible en [kaggle](#). La base de datos contienen información sobre las transacciones hechas los titulares de tarjetas de crédito, durante dos días, en septiembre del 2013, de las cuales 492 son fraudulentas. Este data set es poco balanceado: las clases positivas (fraudulentas) solo corresponden a un 0.172 porcentaje del total de transacciones.

Pregunta 1

Diseña un AE que obtenga pérdida menor o igual a 0.01, tanto en el entrenamiento como la validación. Reporta tu arquitectura.

El Autoencoder que diseñamos para obtener una pérdida menor o igual a 0.01, tanto en el set de entrenamiento como la validación es el siguiente:

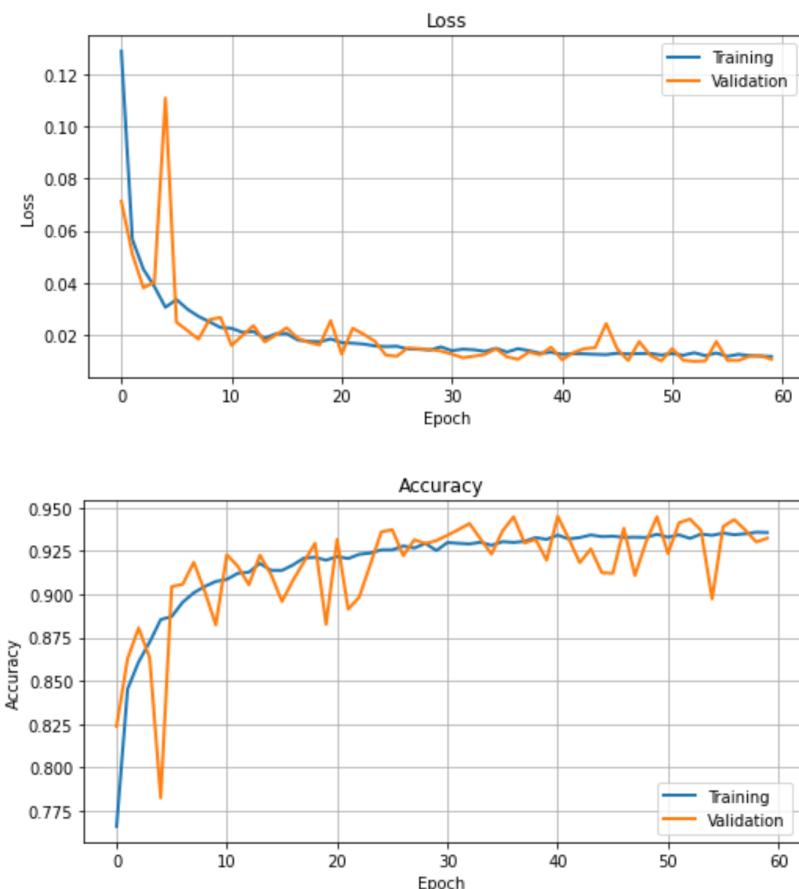
Arquitectura del AE con pérdida menor o igual a 0.01		
Tipo de Capa	Output Shape	Número de Parámetros
Input Layer	(None, 29)	0
Encoder	(None, 500)	15,000
Latent	(None, 16)	8,026
Decoder	(None, 500)	8,500
Output Layer	(None, 29)	14,529

La arquitectura cuenta con:

- > 3 capas ocultas *fully connected*;
- > Cada una de las capas ocultas cuenta con 500, 16, 500 neuronas respectivas.
- > La primera capa oculta sirve como *encoder*; las últimas capas ocultas como *decoder*.
- > La capa latente cuenta con 16 neuronas.
- > El número total de parámetros de la red son: 46,045 parámetros.
- > Para todas las capas se emplearon funciones de activación de tipo "ReLU".
- > Optimizador: Adam.
- > Métrica de pérdida: Error Cuadrático Medio
- > Número de épocas: 60.
- > Tamaño de batch: 32.
- > Pérdida final en el entrenamiento: 0.0116.
- > Pérdida final en la validación: 0.0105.

Las gráficas de pérdida y de precisión de la arquitectura implementada se ven de la siguiente manera:

Modelo: autoencoder con pérdida menor o igual a 0.01



Antes de continuar respondiendo el resto de las preguntas, es necesario revisar los conceptos de los *autoencoders*:

1. Un *autoencoder* es una red neuronal (clasificada como aprendizaje no supervisado), cuyo objeto es generar nuevos datos a partir de los datos de entrada. Los nuevos datos son dispuestos en forma de variables latentes (ubicados en un espacio o capa latente), para posteriormente reconstruir la salida.
2. Este tipo de red está compuesta por dos funciones: un **encoder**, que proyecta los datos dispuestos en una alta dimensionalidad, a una de baja dimensionalidad; **decoder**, que proyecta los datos de una baja dimensionalidad a otra de alta dimensionalidad.
3. La utilidad del entrenamiento del autoencoder va más allá de una simple copia de los datos de entrada en los datos de salida: se espera que la **representación latente**, a través de sus variables latentes, adquiera propiedades útiles que permitan, en este ejercicio, la clasificación de las transacciones en transacciones fraudulentas o transacciones normales.

Pregunta 2

¿Existe alguna relación entre la profundidad del AE y la pérdida final?

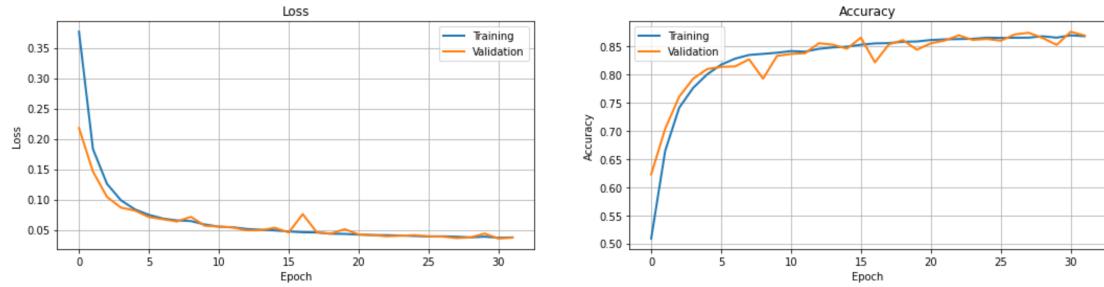
Después de implementar múltiples arquitecturas, identificamos que para, este problema en particular, no hay relación entre la profundidad del *Autoencoder*, y la pérdida final.

Sin embargo, si notamos que la profundidad de la red sí influye en la estabilidad y convergencia hacia una pérdida y precisión final. En este sentido, nuestros mejores resultados fueron aquellos encoders cuyas arquitecturas eran simples (un encoder, una capa latente y una capa decoder).

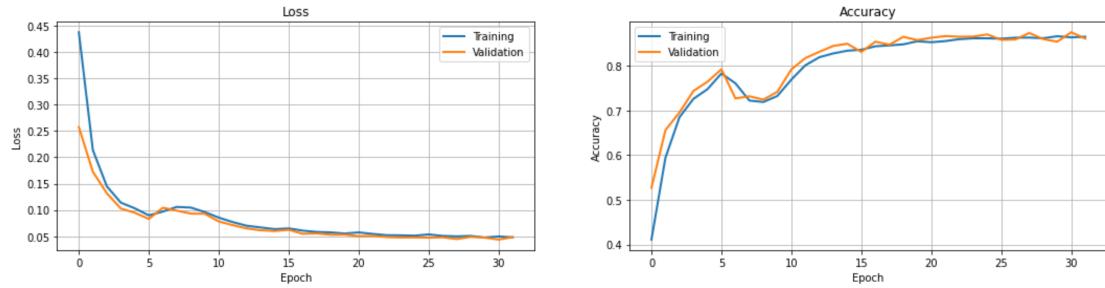
Al contar con este tipo de estructuras simples, fue determinante incorporar un número grande de neuronas en la capas *encoder* y *decoder*, cuidando que la **capa o espacio latente** tenga menos neuronas que las capas de entrada y de salida: de esta manera, nos asegurarnos que la capa latente aprende el mayor número de patrones de los datos de entrada, ignorando los "datos ruidosos".

A continuación, contrastamos algunas arquitecturas que implementamos, junto con sus resultados en términos de pérdida final.

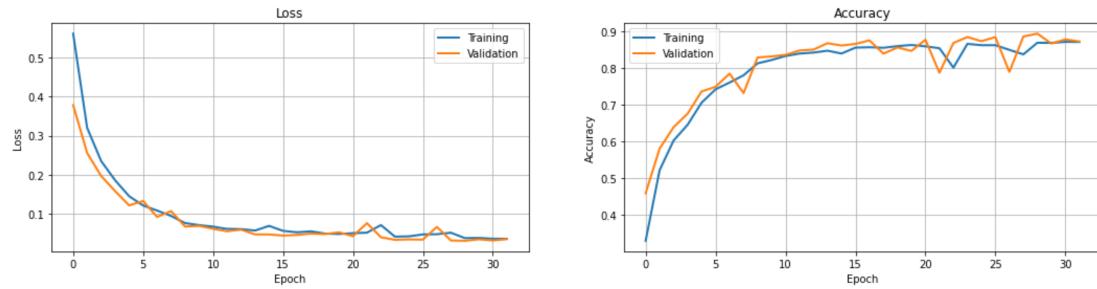
Autoencoder, 3 capas ocultas: un encoder, una capa latente y capa decoders.



Autoencoder, 5 capas ocultas: dos encoders, una capa latente y dos capas decoders.



Autoencoder, 7 capas ocultas: tres encoders, una capa latente y tres capas decoders.

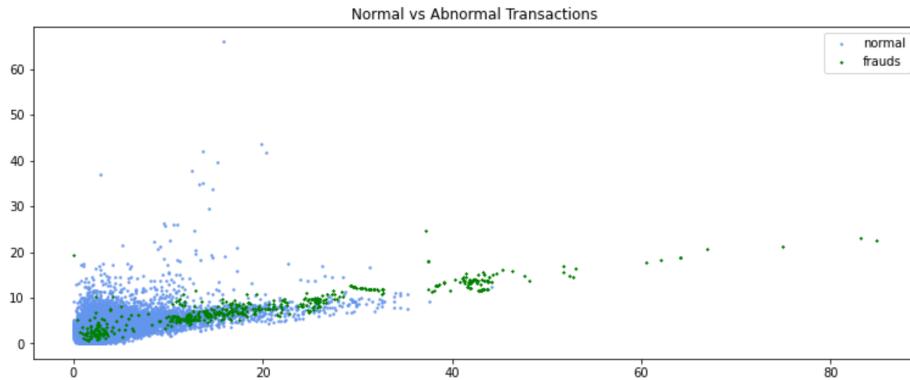


Pregunta 3

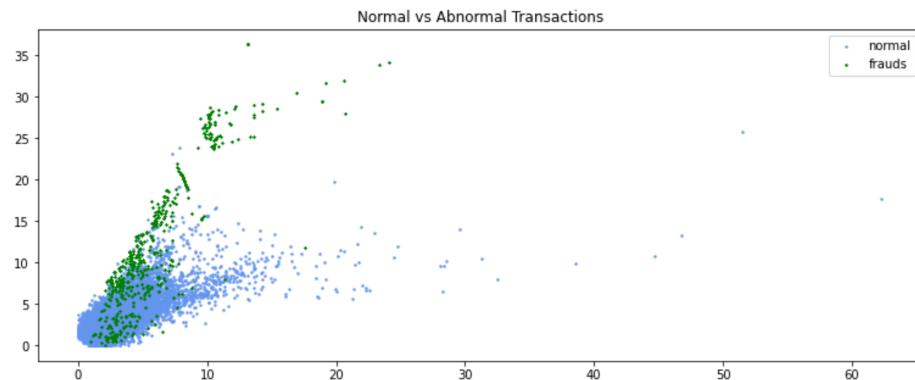
¿Existe alguna relación entre la profundidad del AE y la separación resultante?

En general, la profundidad del AE y la separación resultante no tiene una relación directa. A continuación, se presentan algunas estructuras que se implementarán para evaluar la relación entre profundidad del AE y la separación resultante.

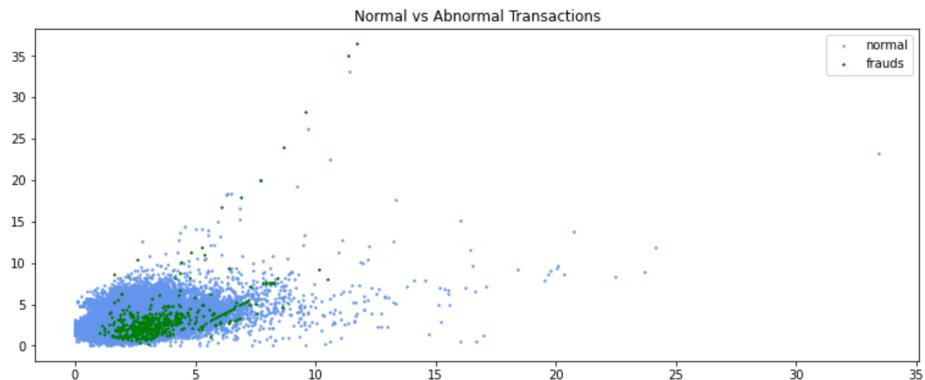
Separación en un plano 2D. Autoencoder, 3 capas ocultas: un encoder, una capa latente y una capa decoders.



Separación en un plano 2D. Autoencoder, 5 capas ocultas: dos encoders, una capa latente y dos capas decoders.

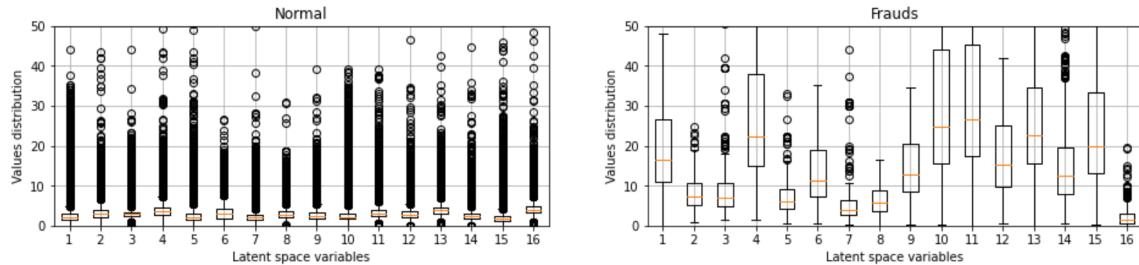


Separación en un plano 2D. Autoencoder, 7 capas ocultas: tres encoders, una capa latente y tres capas decoders.

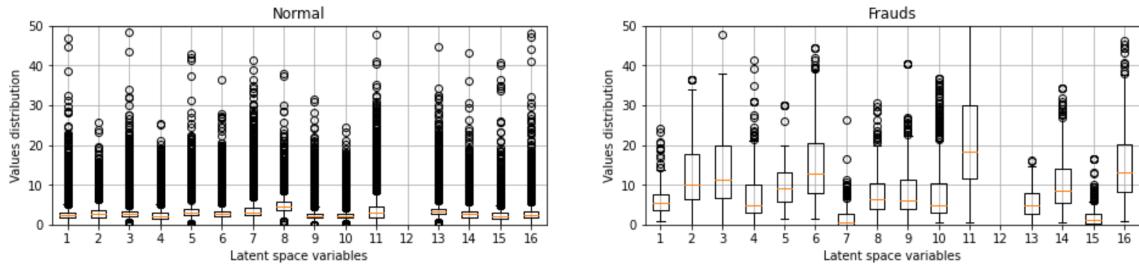


Para este ejemplo en particular, se puede observar que la arquitectura con mayor número de capas (7 capas), es en la presenta menor separación entre las clases de transacciones.

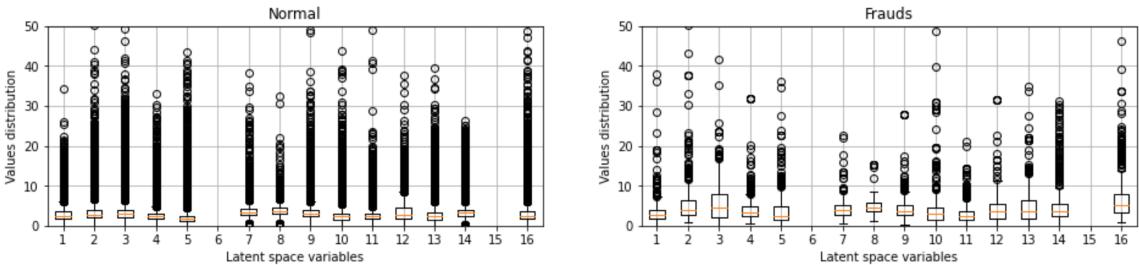
Autoencoder, 3 capas ocultas: un encoder, una capa latente y una capa decoders.



Autoencoder, 5 capas ocultas: dos encoders, una capa latente y dos capas decoders.



Autoencoder, 7 capas ocultas: tres encoders, una capa latente y tres capas decoders.



Por otro lado, en este segundo grupo de gráficas, identificamos que en las arquitecturas con mayor número de capas, el espacio latente tiene más posiciones en las que no se representa ningún valor.

Por lo tanto, considerando los resultados y observaciones contrapuestos en los dos grupos de gráficas que presentamos anteriormente (las arquitecturas con un mayor número de capas, presentan una separación de clases menor, sin embargo, en las arquitecturas con un mayor número de capas existe un mayor número de latent space variables sin valor alguno), **no podemos concluir que hay una relación directa entre profundida del AE y la separación resultante.**

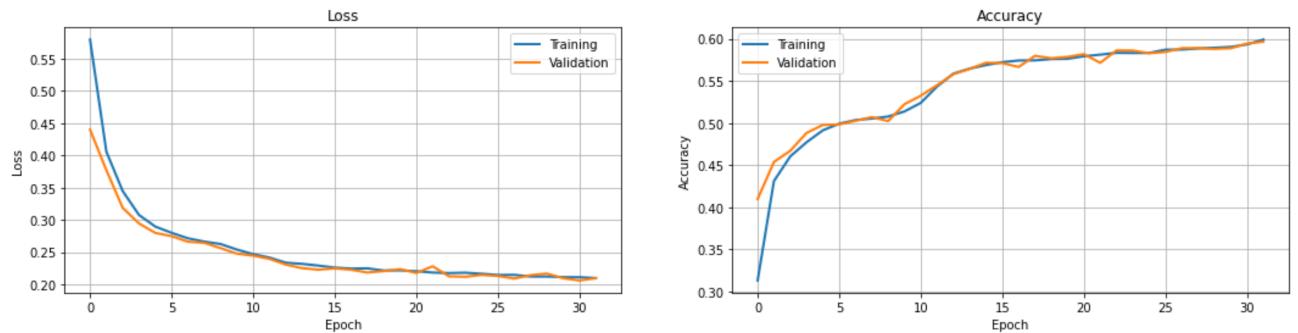
Pregunta 4

¿Existe alguna relación entre el número de elementos en el espacio latente y la pérdida final?

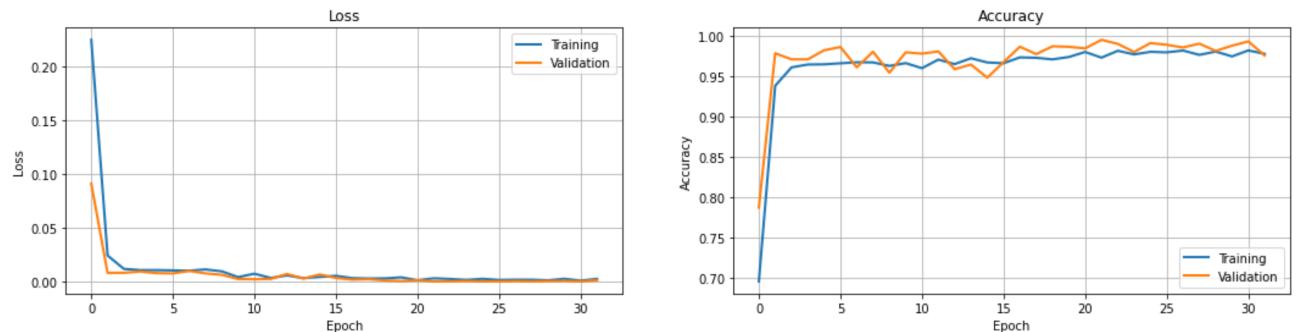
La relación entre el número de elementos en el espacio latente h (neuronas o variables latentes) y pérdida final es inversa: cuanto más elementos hay en el espacio latente, menor pérdida final; al revés, cuanto menos elementos hay en el espacio latente, más pérdida final.

¿Por qué? Al entrenar una representación con menos elementos que la dimensión de los datos de entradas, forzamos al espacio latente h a **aprender las características más relevantes**. Ello se traducirá inevitablemente a una mayor pérdida final, puesto que el trabajo del espacio latente en este caso es más complicado que un simple copiado de información. Si el espacio latente h tiene dimensiones mayores a los datos de entrada, entonces se dota al autoencoder con demasiado capacidad para aprender a realizar la tarea de **copiado de información**, sin extracción alguna de información relevante sobre la distribución de los datos, pero con pérdida final casi nula.

Autoencoder con tres capas ocultas: una capa encoder con 32 neuronas, una latente con 8 neuronas y una capa decoder con 32 neuronas.



Autoencoder con tres capas ocultas: una capa encoder con 32 neuronas, una latente con 29 neuronas y una capa decoder con 32 neuronas.



Pregunta 5

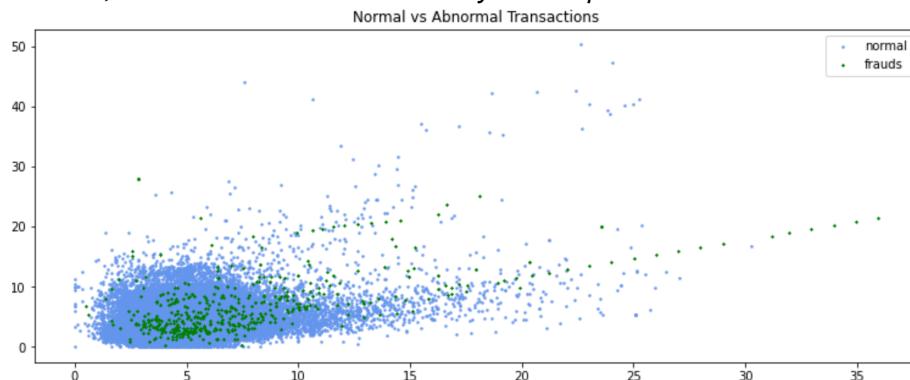
¿Existe alguna relación entre el número de elementos en el espacio latente h y la separación resultante entre los espacios latentes de los datos normales y anormales?

Al igual que en la respuesta anterior, la relación entre el número de elementos en el espacio latente h y la separación resultante de los datos normales y anormales es directa: cuanto menos elementos en el espacio latente, menor capacidad de separación tendrá nuestro autoencoder; cuanto mayor sea el número de elementos en el espacio latente, mayor será la capacidad de separación entre los datos normales y anormales. ¿La explicación es muy similar a la proporcionada en la pregunta 4.

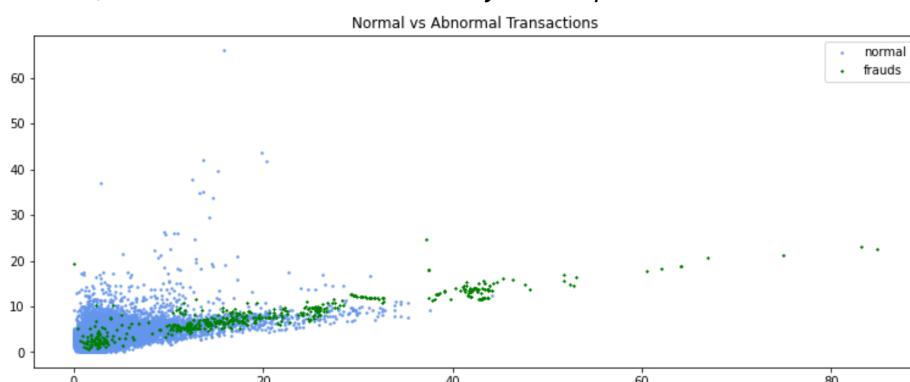
Al entrenar una representación con menos elementos que la dimensión de los datos de

entradas, forzamos al *espacio latente h* a **aprender las características o features más relevantes**, dispuestas en forma de variables latentes, que permitan la separación entre las dos clases de datos: normales y anormales. Como las características importantes se están almacenando en menor número de variables latentes, se ve obligado a comprimir más la información, pudiendo resultar en una menor separación entre las dos clases de datos; al contrario si el espacio latente *h* tiene más elementos, la tarea del autoencoder se reduce a un simple **copiado de información**, sin extracción alguna de información relevante sobre la distribución de los datos, generando variables latentes vacías, que son capaces de separar perfectamente las clases porque simplemente copiaron las características, sin aprender nada.

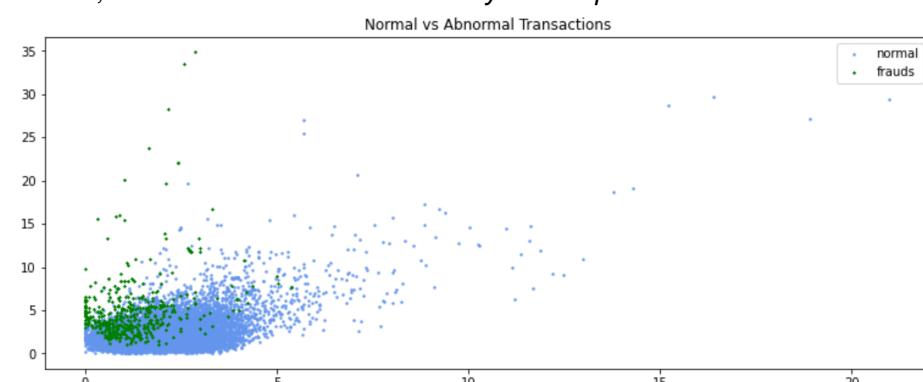
Separación en un plano 2D. Autoencoder con tres capas ocultas: una capa encoder con 32 neuronas, una latente con 8 neuronas y una capa encoder con 32 neuronas.



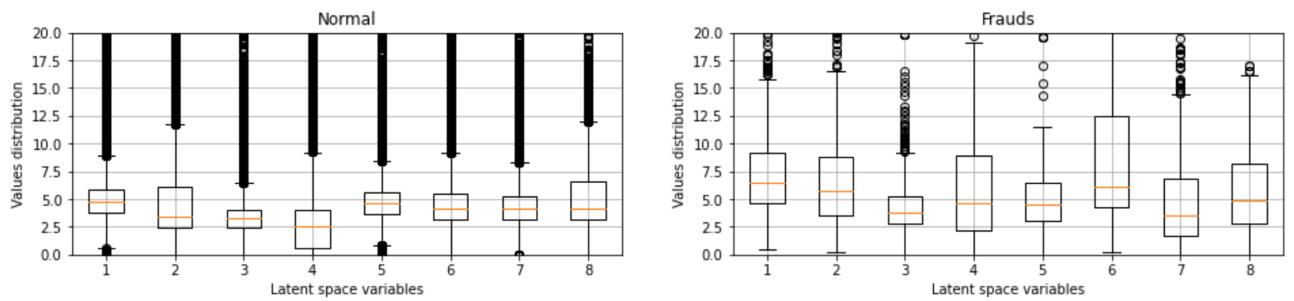
Separación en un plano 2D. Autoencoder con tres capas ocultas: una capa encoder con 32 neuronas, una latente con 16 neuronas y una capa encoder con 32 neuronas.



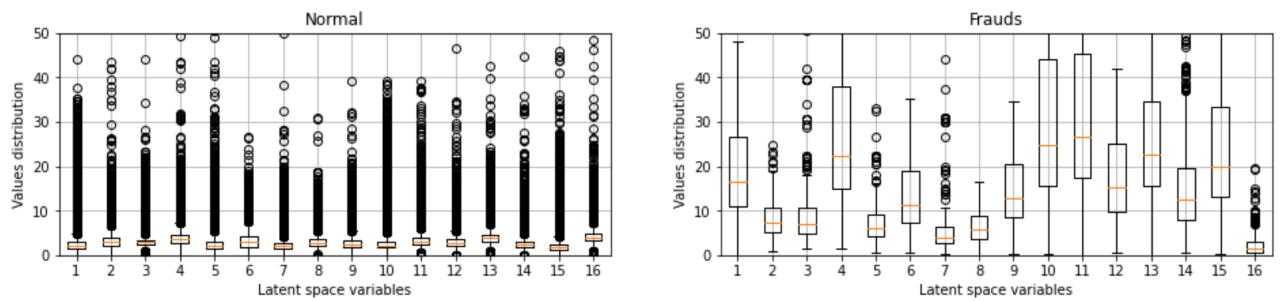
Separación en un plano 2D. Autoencoder con tres capas ocultas: una capa encoder con 32 neuronas, una latente con 29 neuronas y una capa encoder con 32 neuronas.



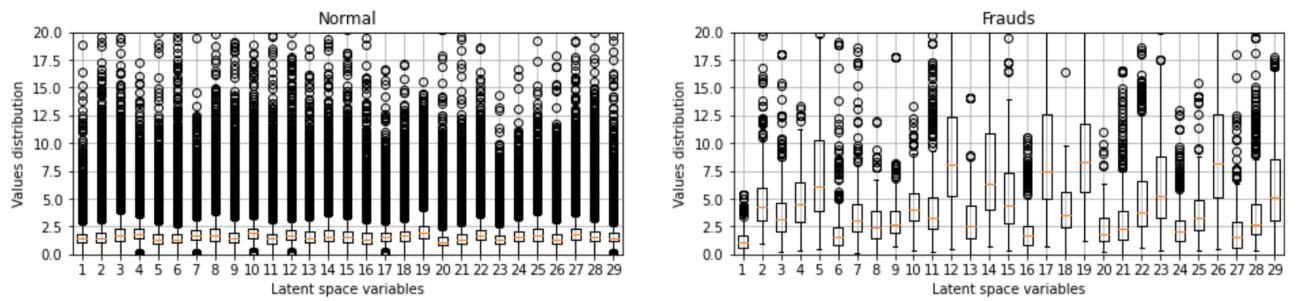
Autoencoder con tres capas ocultas: una capa encoder con 32 neuronas, una latente con 8 neuronas y una capa encoder con 32 neuronas.



Autoencoder con tres capas ocultas: una capa encoder con 32 neuronas, una latente con 16 neuronas y una capa encoder con 32 neuronas.



Autoencoder con tres capas ocultas: una capa encoder con 32 neuronas, una latente con 29 neuronas y una capa encoder con 32 neuronas.



Por lo observado en los resultados de esta pregunta y las anteriores, podemos afirmar que uno de los trabajos principales del espacio latente h es la reducción de la dimensionalidad de los datos de entrada. Una vez identificados los patrones principales de los datos (en general), los *outliers* serán revelados. Así, durante la implementación de los distintos de modelos realizados para la entrega del presente trabajo, la evaluación de la bondad de nuestros modelos fue desafiante: un constante balanceo entre nuestra pérdida final y la separación de los datos.

Pregunta 6

Intenta forzar el espacio latente para que sea ralo. Reporta tu mejor modelo y el desempeño que obtenga, tanto en pérdida como en capacidad de diferenciar datos anormales.

Una manera de restringir las representaciones para que sean compactas es añadiendo una restricción adicional al espacio latente: que sea ralo. Una manera de forzar lo anterior es añadiendo regularizadores en esa capa, para que menos unidades sean activadas durante cierto tiempo. Para ello, se implementó el siguiente código:

Listing 1: Autoencoder con Espacio Latente de Tipo Ralo

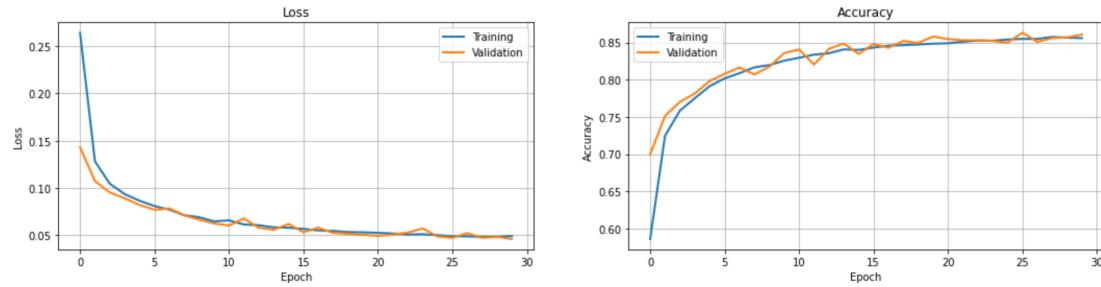
```
1 # Create model >0.01
2 in_layer = Input(shape=(x_train.shape[1],))
3 encoder = Dense(128, activation='relu')(in_layer)
4 latent = Dense(12, activation='relu', activity_regularizer=←
    regularizers.l1(10e-5))(encoder)
5 decoder = Dense(128, activation='relu')(latent)
6 out_layer = Dense(x_train.shape[1])(decoder)
7 AE = Model(inputs=in_layer, outputs=out_layer)
8 AE.summary()
```

Los hiperparámetros definidos fueron los siguientes:

- > Optimizador: Adam.
- > Métrica de Pérdida: Error Cuadrático Medio.
- > Batch size: 64.
- > Número de épocas: 30.

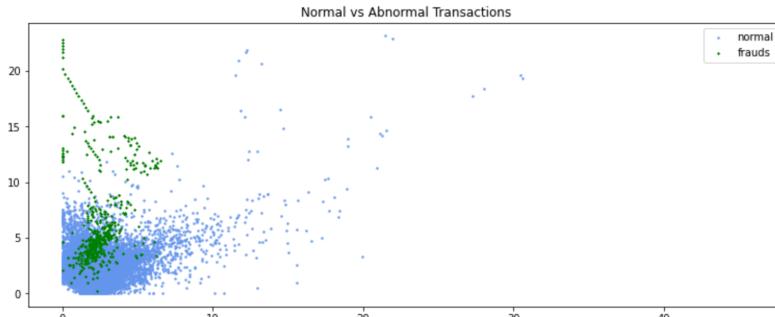
Los resultados en términos de pérdida final y capacidad de diferenciar los datos anormales, se pueden identificar a través de las siguientes gráficas:

Pérdida y Precisión: Autoencoder con tres capas ocultas: una capa encoder con 128 neuronas, una latente con 12 neuronas y una capa encoder con 128 neuronas.

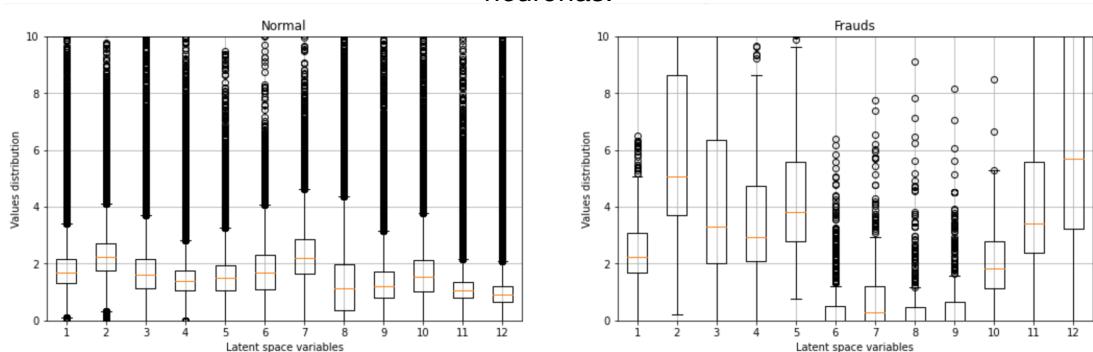


visiónes a los modelos no importantes

Separación en un plano 2D: Autoencoder con tres capas ocultas: una capa encoder con 128 neuronas, una latente con 12 neuronas y una capa encoder con 128 neuronas.



Distribución de los valores en el espacio latente h : Autoencoder con tres capas ocultas: una capa encoder con 128 neuronas, una latente con 12 neuronas y una capa encoder con 128 neuronas.



Referencias

- A wizard's guide to Adversarial Autoencoders: Part 2, Exploring latent space with Adversarial
- Deep Inside: Autoencoders
- Comprehensive Introduction to Autoencoders
- Towards an Interpretable Latent Space