

Redes Neuronales III - keras

Jonnatan Arias Garcia – jonnatan.arias@utp.edu.co

David Cardenas peña - dcardenasp@utp.edu.co

Hernán Felipe Garcia - hernanf.garcia@udea.edu.co

Contenido

Módulos y funciones útiles

- ☐ Conversión de datos categóricos
- ☐ Print de keras
- ☐ Guardar modelos

Interpretación de loss curve

- ☐ + AUC

Topics extra:

- ☐ Transfer learning
- ☐ Otras librerías

Colab.

Data Augmentation

CNN.

Funciones útiles

print_summary

Muestra el resumen del modelo

```
from keras.utils import print_summary
```

```
print_summary(model)
```

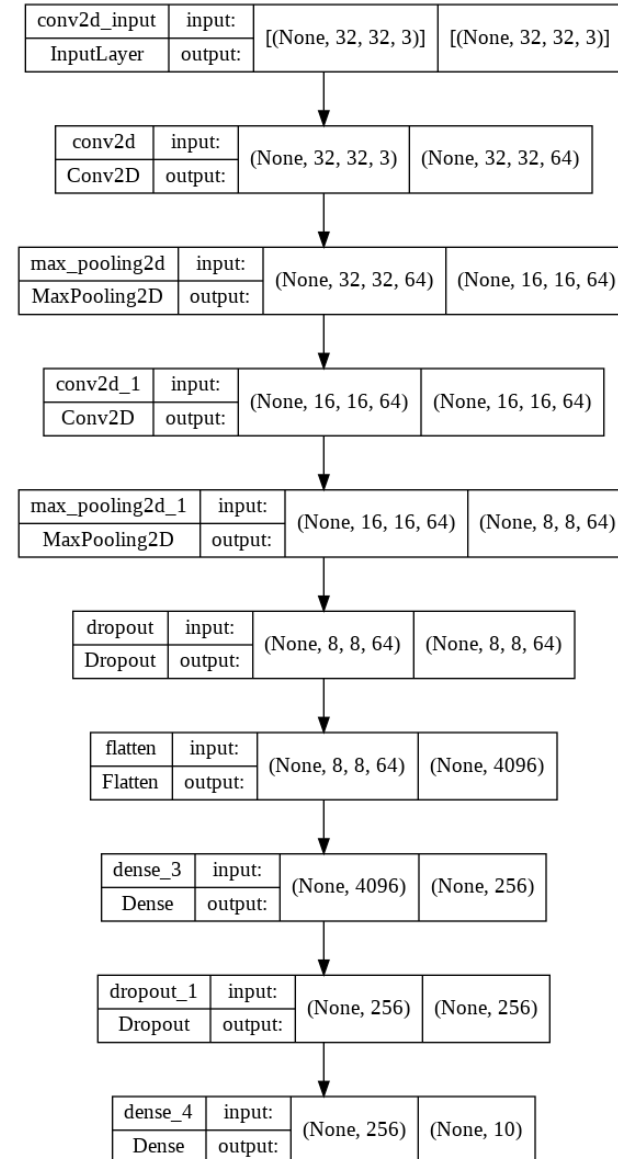
```
Model: "sequential"
-----
Layer (type)                 Output Shape              Param #
-----
conv2d (Conv2D)              (None, 30, 30, 32)        896
-----
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)         0
-----
conv2d_1 (Conv2D)            (None, 13, 13, 64)       18496
-----
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
-----
conv2d_2 (Conv2D)            (None, 4, 4, 64)         36928
-----
flatten (Flatten)            (None, 1024)              0
-----
dense (Dense)                (None, 64)                65600
-----
dense_1 (Dense)              (None, 10)                 650
-----
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
-----
```

Funciones útiles

plot_model

Muestra la representación del modelo en formato punto y permite guardarlo

```
from keras.utils import plot_model
plot_model(model, to_file=image.png)
```



Funciones útiles

To_categorical

Convierte vector de clase a matriz de clase binaria (one hot encoding)

```
>>> from keras.utils import to_categorical
>>> labels = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> to_categorical(labels)
array([[1., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 1., 0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 1., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 0., 0., 0., 0., 1.]], dtype = float32)
```

Funciones útiles

EarlyStopping: **Callback** que detiene el entrenamiento cuando una **métrica** monitoreada deja de mejorar.

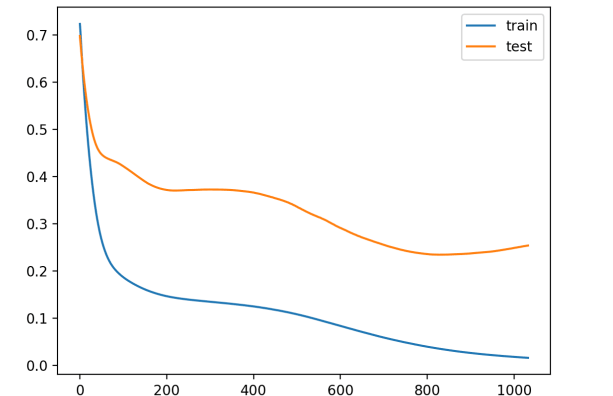
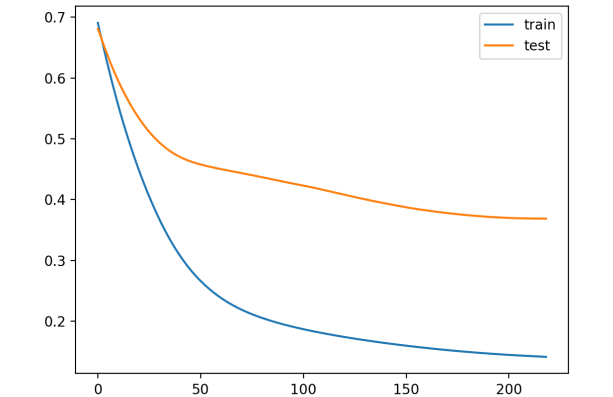
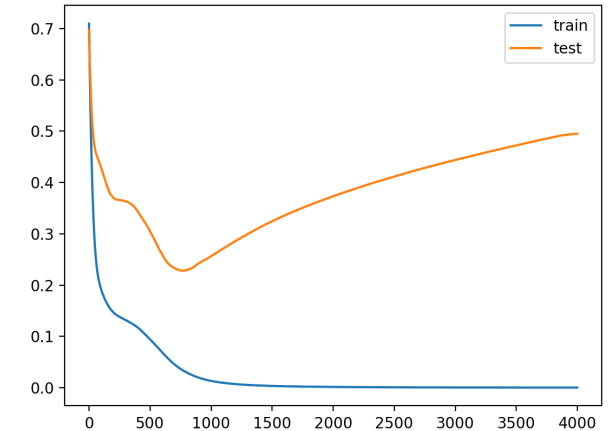
patience: numero de épocas sin mejora

```
from keras.callbacks import EarlyStopping
```

```
early_stopping = EarlyStopping(monitor='val_loss', patience=3)
```

```
model.fit(x_train, y_train, validation_data=(x_val, y_val),  
callbacks=[early_stopping])
```

Ultimo con mas patiences=200



Funciones útiles

Guardar Checkpoints

Para guardar el modelo o los pesos del modelo con cierta frecuencia.

Se usa junto al entrenamiento para guardar un modelo o pesos (en un archivo) en algún intervalo, de modo que posteriormente podamos cargarlos para continuar con el entrenamiento desde una punto dado.

```
keras.callbacks.ModelCheckpoint(  
    filepath,  
    monitor="val_loss",  
    verbose=0,  
    save_best_only=False,  
    save_weights_only=False,  
    mode="auto",  
    save_freq="epoch",  
    initial_value_threshold=None,  
)
```

Funciones útiles

Algunas opciones que ofrece esta devolución de llamada incluyen:

- Ya sea para conservar únicamente el modelo que ha logrado el "mejor rendimiento" hasta el momento o para guardar el modelo al final de cada época, independientemente del rendimiento.
- Definición de "mejor"; qué cantidad **monitorear** y si se debe maximizar o minimizar.
- La frecuencia con la que debería guardarse. Actualmente, la devolución de llamada admite guardar al final de cada **época** o después de una cantidad fija de lotes de entrenamiento.
- Si solo se guardan los **pesos** o se guarda **todo** el modelo.
- La opción **mode** 'auto' puede ser max o min dependiendo la métrica, loss='min' accuracy='max'

```
keras.callbacks.ModelCheckpoint(  
    filepath,  
    monitor="val_loss",  
    verbose=0,  
    save_best_only=False,  
    save_weights_only=False,  
    mode="auto",  
    save_freq="epoch",  
    initial_value_threshold=None,  
)
```


Funciones útiles

Checkpoint: **Callback** que carga los datos del modelo incluyendo los pesos (en la dirección y archivo 'C:\Mis documento\best_model.h5')

```
from keras.callbacks import ModelCheckpoint
```

```
model_checkpoint = ModelCheckpoint('best_model.h5', monitor='val_accuracy',  
save_best_only=True)
```

```
model.fit(x_train, y_train, validation_data=(x_val, y_val),  
callbacks=[model_checkpoint])
```

.

Funciones útiles

Igualmente podemos cargar un modelo pre-entrenado para predecir.

```
model = create_model()
```

```
keras.models.load_model('best_model.h5')
```

O solo lo pesos

```
model.load_weights('best_model.h5')
```

Quiero guardar mi modelo entrenado como un zip

```
model.save('my_model.keras')
```

Y si quiero cargarlos

```
new_model = tf.keras.models.load_model('my_model.keras')
```

.

Loss Curve

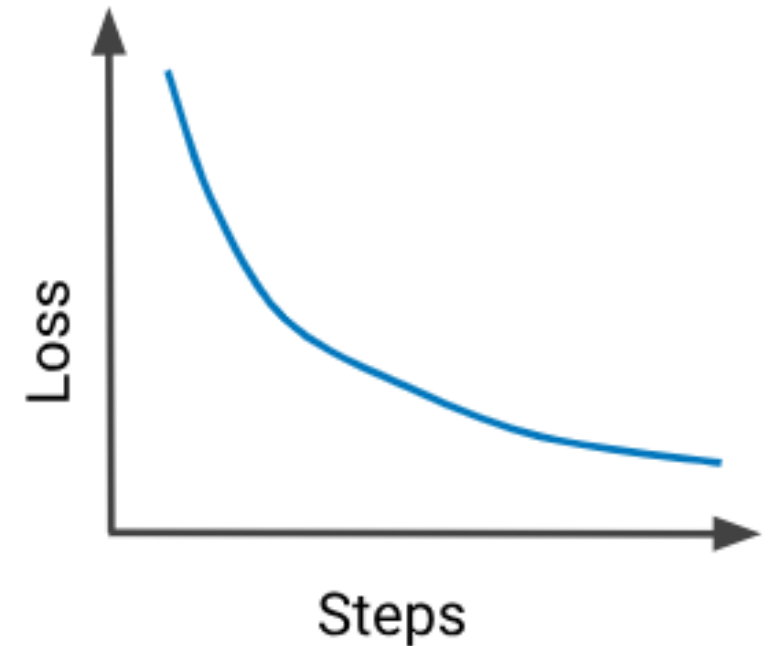
Seria genial que todas las curvas de perdida se vieran así:

Pero, en realidad, las curvas de pérdida pueden ser bastante difíciles de interpretar.

<https://developers.google.com/machine-learning/testing-debugging/metrics/interpretic?hl=es-419>

Curso:

<https://developers.google.com/machine-learning?hl=es-419>

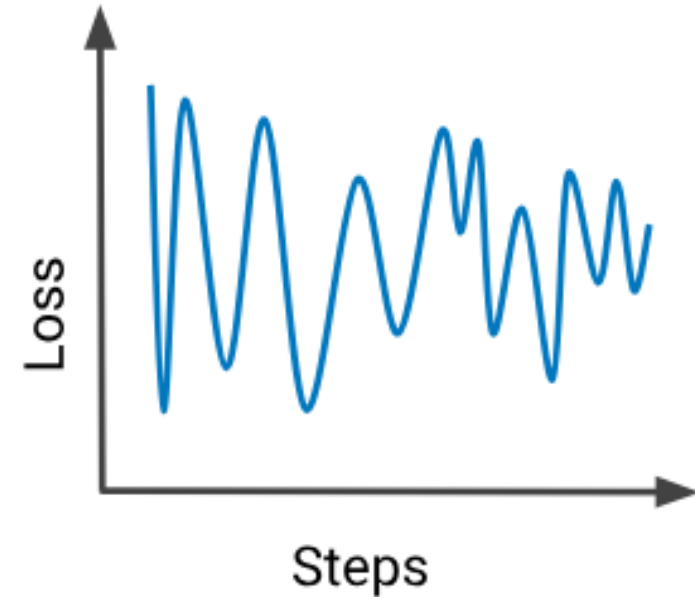


Loss Curve

1. ¡Mi modelo no se entrena!

Tu modelo no está convergiendo. Prueba esto:

- Parece inestable: reduce la tasa de aprendizaje para evitar que el modelo rebote en el espacio de parámetros.
- Simplifica tu conjunto de datos: Datos manejables en testeo y asegúrate que sabes que tu modelo puede predecir.
- Simplifica tu modelo y asegúrate de que el rendimiento supere el modelo de referencia. Luego, agrega complejidad al modelo de manera gradual
- Verifica que los datos estén en los formatos correctos y las etiquetas sean adecuadas de la base de datos
- Utilizar un esquema de validación adecuado. Puede ser por batch



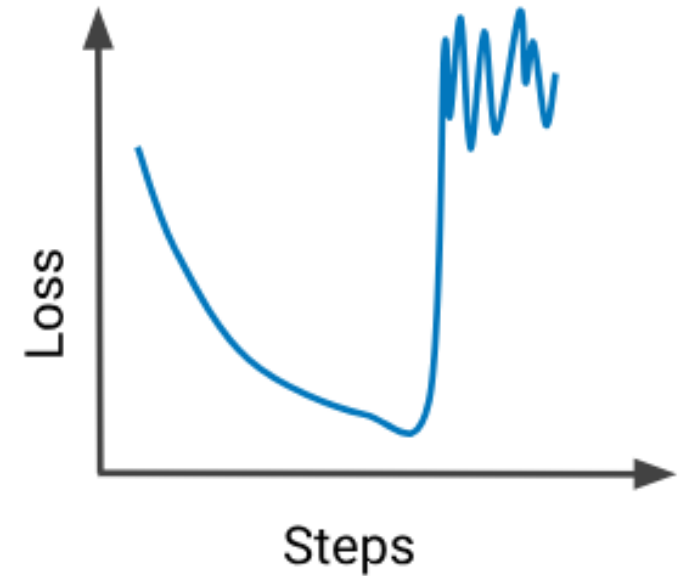
Loss Curve

2. ¡Explotó mi pérdida!

Por lo general se da por valores anómalos en los datos de entrada.

- NaN en los datos de entrada
- Gradiente con alto crecimiento debido a datos anómalos
- División por cero
- Logaritmo de cero o numero negativos

Para corregir una pérdida con alto crecimiento, verifica si hay datos anómalos en los datos. Si la anomalía parece problemática, investiga la causa. De lo contrario, si la anomalía se parece a los datos periféricos, asegúrate de distribuir los valores atípicos de manera uniforme entre los lotes mediante la redistribución de los datos.



Loss Curve

3. ¡Mis métricas son contradictorias!

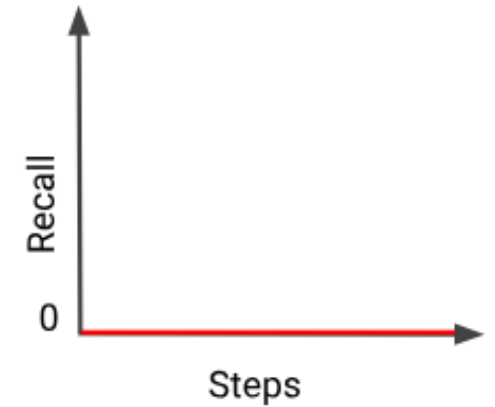
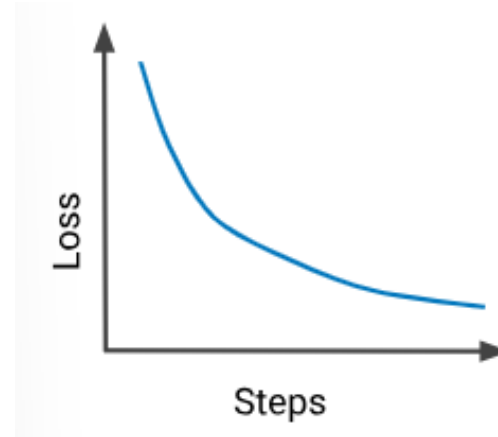
La recuperación está atascada en 0 porque la probabilidad de clasificación de tus ejemplos nunca es superior al umbral para una clasificación positiva.

Esta situación suele ocurrir en problemas con un gran desequilibrio de clases.

Nota: Las bibliotecas como TF Keras, suelen usar un umbral predeterminado de 0.5 para calcular las métricas de clasificación.

Prueba estos pasos:

- Disminuye el umbral de clasificación.
- Verifica las métricas que no varían, como el **AUC** (área bajo la curva)

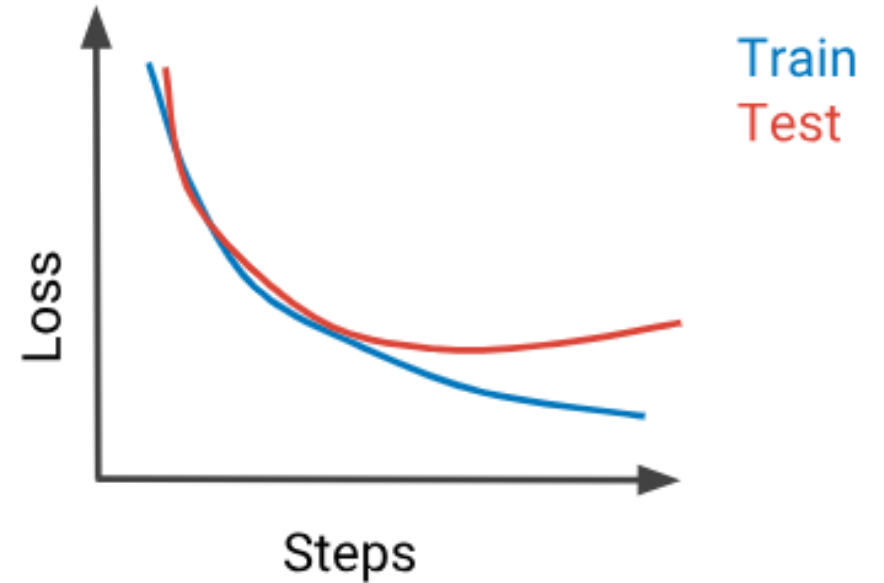


Loss Curve

4. ¡La pérdida de prueba es muy alta!

Tu modelo se sobreajusta a los datos de entrenamiento.

- Reduce la capacidad del modelo.
- Agrega regularización.
- Verifica que las divisiones de entrenamiento y de prueba sean equivalentes en términos estadísticos.

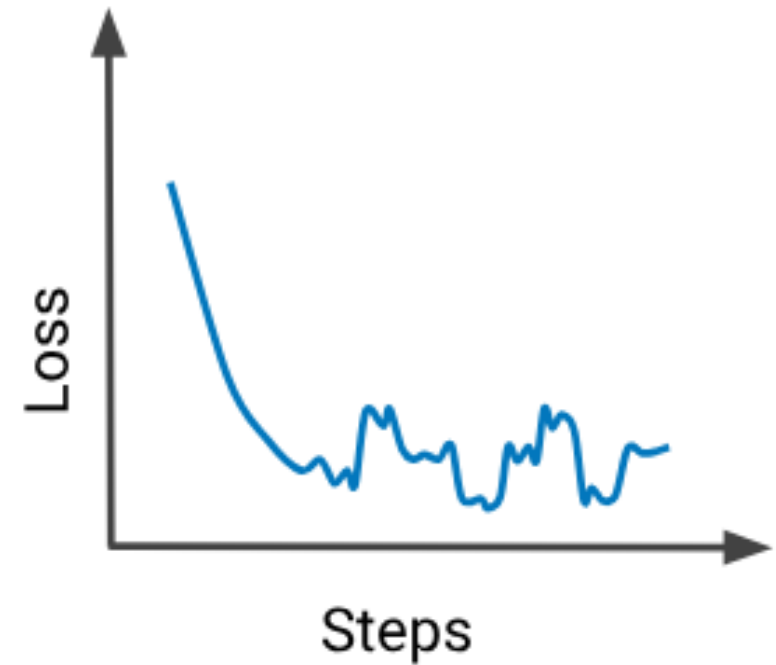


Loss Curve

5. Mi modelo se estanca

Su pérdida muestra un comportamiento repetitivo similar a un paso.

Es probable que los datos de entrada que ve tu modelo tengan un comportamiento repetitivo. Asegúrate de que la redistribución quite el comportamiento repetitivo de los datos de entrada.



Inciso AUC I

Medida agregada del rendimiento en clasificación.

El AUC es como la probabilidad de que el modelo clasifique un ejemplo positivo aleatorio más alto que un ejemplo negativo aleatorio.

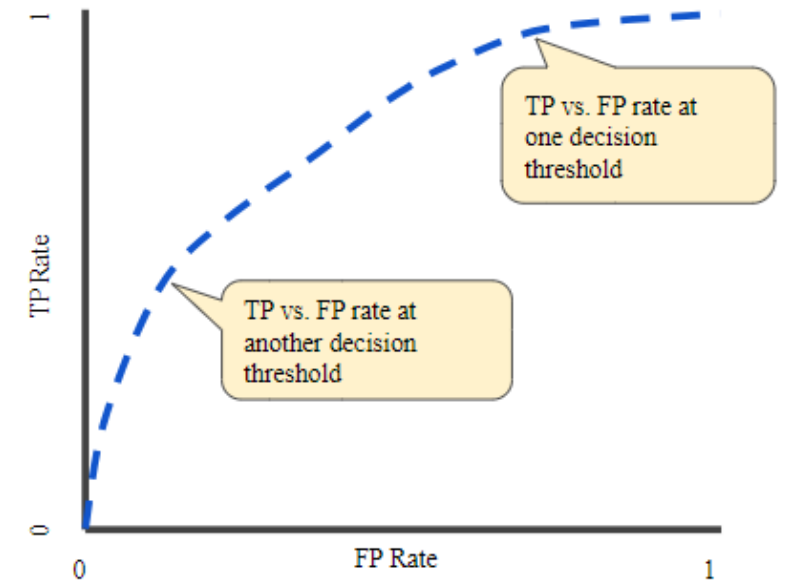
Tasa de verdaderos positivos (TPR) es sinónimo de exhaustividad y, por lo tanto, se define de la siguiente manera:

$$TPR = \frac{TP}{TP + FN}$$

Tasa de falsos positivos (FPR) se define de la siguiente manera:

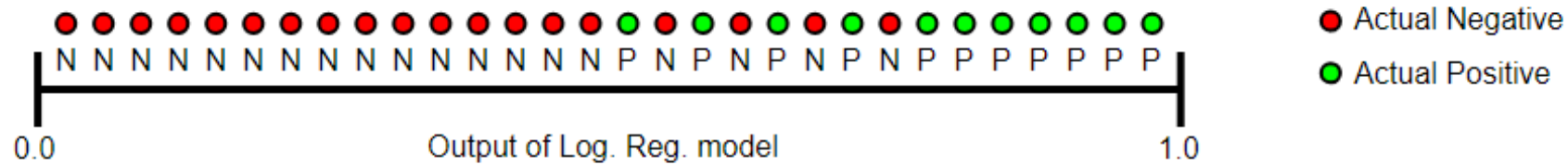
$$FPR = \frac{FP}{FP + TN}$$

Esperaríamos ver una curva exponencial ascendente hasta a 1, el optimo seria un escalón 1 desde x=0



Inciso AUC II

En el siguiente ejemplo, Se ordenan de izquierda a derecha en orden ascendente con respecto a las predicciones de regresión logística:



- El AUC representa la probabilidad de que un ejemplo aleatorio positivo (verde) se posicione a la derecha de un ejemplo aleatorio negativo (rojo).
- El AUC varía en valor de 0 a 1. Un modelo cuyas predicciones son un 100% **incorrectas** tiene un AUC de **0.0**; uno cuyas predicciones son un 100% **correctas** tiene un AUC de **1.0**.

El AUC es conveniente por los siguientes dos motivos:

- El AUC es invariable con respecto a la escala. Mide qué tan bien se clasifican las predicciones, en lugar de sus valores absolutos.
- El AUC es invariable con respecto al umbral de clasificación. Mide la calidad de las predicciones del modelo, independientemente del umbral de clasificación elegido.

Transfer Learning

Aprendizaje por transferencia, es una donde se aprovecha el conocimiento adquirido al resolver una tarea para resolver otra tarea relacionada.

Implica tomar un modelo **pre-entrenado** en un conjunto de datos específico y ajustarlo para resolver una tarea diferente.




Cuando un modelo ya esta muy afinado, lo dejamos congelado y adicionamos capaz sin modificar la red ni pesos originales

El transfer learning es beneficioso por varias razones:

- **Reducción del tiempo de entrenamiento**
- **Necesidad de datos reducida**
- **Mejora del rendimiento**

Se utiliza comúnmente en tareas como clasificación de imágenes, detección de objetos, procesamiento de lenguaje natural (NLP) y muchas otras áreas donde los conjuntos de datos pueden ser limitados y costosos de recopilar.

Keras Vs.

	Keras 	TensorFlow 	PyTorch 
Level of API	high-level API ¹	Both high & low level APIs	Lower-level API ²
Speed	Slow	High	High
Architecture	Simple, more readable and concise	Not very easy to use	Complex ³
Debugging	No need to debug	Difficult to debugging	Good debugging capabilities
Dataset Compatibility	Slow & Small	Fast speed & large	Fast speed & large datasets
Popularity Rank	1	2	3
Uniqueness	Multiple back-end support	Object Detection Functionality	Flexibility & Short Training Duration
Created By	Not a library on its own	Created by Google	Created by Facebook ⁴
Ease of use	User-friendly	Incomprehensive API	Integrated with Python language
Computational graphs used	Static graphs	Static graphs	Dynamic computation graphs ⁵