taller_twitter

April 5, 2022

1 Taller 1: Aprendizaje Profundo

1.0.1 Profesor: Ing. Julio Omar Palacio Niño

1.0.2 Estudiante: Jesús Ernesto Suárez Triana

El objetivo del siguiente taller es hacer un análisis de clasificación empleando redes neuronales feed-forward

1.1 1. Comprensión del Dataset

Se trabajará con el dataset de la página Kaggle, por lo cual deberá ser descargado del siguiente enlace link_kaggle_dataset

```
Info Dataset
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 26 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   _unit_id               5000 non-null   int64
 1   _golden                5000 non-null   bool
 2   _unit_state            5000 non-null   object
 3   _trusted_judgments     5000 non-null   int64
 4   _last_judgment_at      5000 non-null   object
 5   gender                 4964 non-null   object
 6   gender:confidence      4991 non-null   float64
 7   profile_yn             5000 non-null   object
 8   profile_yn:confidence  5000 non-null   float64
 9   created                5000 non-null   object
 10  description            4329 non-null   object
 11  fav_number             5000 non-null   int64
 12  gender_gold            0 non-null      float64
 13  link_color             5000 non-null   object
 14  name                   5000 non-null   object
 15  profile_yn_gold        0 non-null      float64
 16  profileimage           5000 non-null   object
 17  retweet_count          5000 non-null   int64
 18  sidebar_color          5000 non-null   object
 19  text                   5000 non-null   object
```

```
 20  tweet_coord            53 non-null     object
 21  tweet_count            5000 non-null   int64
 22  tweet_created          5000 non-null   object
 23  tweet_id               5000 non-null   float64
 24  tweet_location         3338 non-null   object
 25  user_timezone          3265 non-null   object
dtypes: bool(1), float64(5), int64(5), object(15)
memory usage: 981.6+ KB
None
```

[6]:

	_unit_id	_golden	_unit_state	_trusted_judgments	_last_judgment_at
0	815719226	False	finalized	3	10/26/15 23:24
1	815719227	False	finalized	3	10/26/15 23:30
2	815719228	False	finalized	3	10/26/15 23:33
3	815719229	False	finalized	3	10/26/15 23:10
4	815719230	False	finalized	3	10/27/15 1:15

	gender	gender:confidence	profile_yn	profile_yn:confidence
0	male	1.0000	yes	1.0
1	male	1.0000	yes	1.0
2	male	0.6625	yes	1.0
3	male	1.0000	yes	1.0
4	female	1.0000	yes	1.0

	created	…	profileimage
0	12/5/13 1:48	…	https://pbs.twimg.com/profile_images/414342229…
1	10/1/12 13:51	…	https://pbs.twimg.com/profile_images/539604221…
2	11/28/14 11:30	…	https://pbs.twimg.com/profile_images/657330418…
3	6/11/09 22:39	…	https://pbs.twimg.com/profile_images/259703936…
4	4/16/14 13:23	…	https://pbs.twimg.com/profile_images/564094871…

	retweet_count	sidebar_color
0	0	FFFFFF
1	0	C0DEED
2	1	C0DEED
3	0	C0DEED
4	0	0

	text	tweet_coord	tweet_count
0	Robbie E Responds To Critics After Win Against…	NaN	110964
1	ÛÏIt felt like they were my friends and I was…	NaN	7471
2	i absolutely adore when louis starts the songs…	NaN	5617
3	Hi @JordanSpieth - Looking at the url - do you…	NaN	1693
4	Watching Neighbours on Sky+ catching up with t…	NaN	31462

	tweet_created	tweet_id	tweet_location	user_timezone
0	10/26/15 12:40	6.587300e+17	main; @Kan1shk3	Chennai

```
1  10/26/15 12:40  6.587300e+17              NaN  Eastern Time (US & Canada)
2  10/26/15 12:40  6.587300e+17           clcncl                    Belgrade
3  10/26/15 12:40  6.587300e+17    Palo Alto, CA  Pacific Time (US & Canada)
4  10/26/15 12:40  6.587300e+17              NaN                         NaN

[5 rows x 26 columns]
```

1.2 2. Comprensión del dataset

- ¿Qué información presenta el dataset?, describir las principales características del dataset, y a su vez identificar la variable sobre la cual se desea hacer la predicción.

Respuesta

Acorde a la página de Kaggle, este es un dataset construido para *CrowdFlower*. Para el, se pidio a los colaboradores ver un perfil de Twitter y decidir si el usuario era un hombre, una mujer o una marca (no individual).

El dataset esta compuesto por 20050 registros y 26 filas.

El diccionario de datos es el siguiente:

- **unitid:** un identificador único para el usuario
- **_golden:** si el usuario se incluyó en el estándar de oro para el modelo; TRUE o FALSE
- **unitstate:** estado de la observación; uno de finalizado (para las observaciones juzgadas por los colaboradores) o dorado (para las observaciones del patrón oro)
- **trustedjudgments:** número de juicios de confianza (int); siempre 3 para las observaciones no doradas, y lo que puede ser un identificador único para las observaciones del patrón oro
- **lastjudgment_at:** fecha y hora de la última sentencia del contribuyente; en blanco para las observaciones gold standard
- **gender:** uno de los siguientes: hombre, mujer o marca (para perfiles no humanos)
- **gender:confidence:** un flotador que representa la confianza en el género proporcionado
- **profile_yn:** "no" parece significar que el perfil debía formar parte del conjunto de datos pero no estaba disponible cuando los colaboradores fueron a juzgarlo
- **profile_yn:confidence:** confianza en la existencia/no existencia del perfil
- **created:** fecha y hora de creación del perfil
- **description:** descripción del perfil del usuario
- **fav_number:** número de tuits que el usuario ha favorecido
- **gender_gold:** si el perfil es dorado, ¿cuál es el género?
- **link_color:** el color del enlace en el perfil, como valor hexadecimal
- **name:** el nombre del usuario
- **profileyngold:** si el valor y/n del perfil es dorado
- **profileimage:** un enlace a la imagen del perfil
- **retweet_count:** número de veces que el usuario ha retuiteado (o posiblemente, ha sido retuiteado)
- **sidebar_color:** color de la barra lateral del perfil, como valor hexadecimal
- **text:** texto de uno de los tweets del usuario al azar
- **tweet_coord:** si el usuario tiene activada la localización, las coordenadas como una cadena con el formato "[latitud, longitud]"
- **tweet_count:** número de tweets que el usuario ha publicado

- **tweet_created:** cuándo se creó el tweet aleatorio (en la columna de texto)
- **tweet_id:** el id del tweet aleatorio
- **tweet_location:** ubicación del tweet; parece que no está especialmente normalizado
- **user_timezone:** la zona horaria del usuario

De la vista previa del Dataset, podemos entender que el dataset tiene como unidad de medida usuarios unicos y las caracteristicas, corresponden a información general asociada a la cuenta del usuario.

También, para el objetivo de clasificación de registros basados en el genero, podemos identificar que la variable target en nuestro caso es **gender**

Como caracteristicas generales del dataset, observamos que: - Tenemos 17 campos string, 5 enteros, 3 flotante y 1 booleano. - De las variables númericas resaltamos la importancia de las variables *fav_number, retweet_count* (No contamos las demás, por naturaleza de la obtención de la data y en particular *gender:confidence* hace relación a la obtención del genero, pero en si no genera una relación al genero en si.) - De las variables categoricas y teniendo en cuenta el objetivo del ejercicio, consideramos importantes las variables de *link_color y sidebar_color* - **nota** No incluimos las variables de texto, y realizar un análisis de sentimiento previo, con el fín de simplificar el ejercicio.

Conteo de Tipos de datos

[7]:
```
object     15
int64       5
float64     5
bool        1
dtype: int64
```

Univariado variables númericas (Entero y Flotante)

[8]:

	_unit_id	_trusted_judgments	gender:confidence	\
count	5.000000e+03	5000.0	4991.000000	
mean	8.157218e+08	3.0	0.887540	
std	1.471170e+03	0.0	0.192198	
min	8.157192e+08	3.0	0.000000	
25%	8.157205e+08	3.0	0.679200	
50%	8.157218e+08	3.0	1.000000	
75%	8.157230e+08	3.0	1.000000	
max	8.157243e+08	3.0	1.000000	

	profile_yn:confidence	fav_number	gender_gold	profile_yn_gold	\
count	5000.000000	5000.000000	0.0	0.0	
mean	0.991837	4443.594600	NaN	NaN	
std	0.051670	12879.525741	NaN	NaN	
min	0.632300	0.000000	NaN	NaN	
25%	1.000000	24.000000	NaN	NaN	
50%	1.000000	504.500000	NaN	NaN	
75%	1.000000	3385.000000	NaN	NaN	
max	1.000000	341621.000000	NaN	NaN	

```
       retweet_count   tweet_count      tweet_id
count    5000.000000  5.000000e+03  5.000000e+03
mean        0.101200  3.774460e+04  6.587300e+17
std         4.679587  1.249624e+05  4.224422e+04
min         0.000000  1.000000e+00  6.587300e+17
25%         0.000000  2.219750e+03  6.587300e+17
50%         0.000000  9.538000e+03  6.587300e+17
75%         0.000000  3.599825e+04  6.587300e+17
max       330.000000  2.372591e+06  6.587300e+17
```

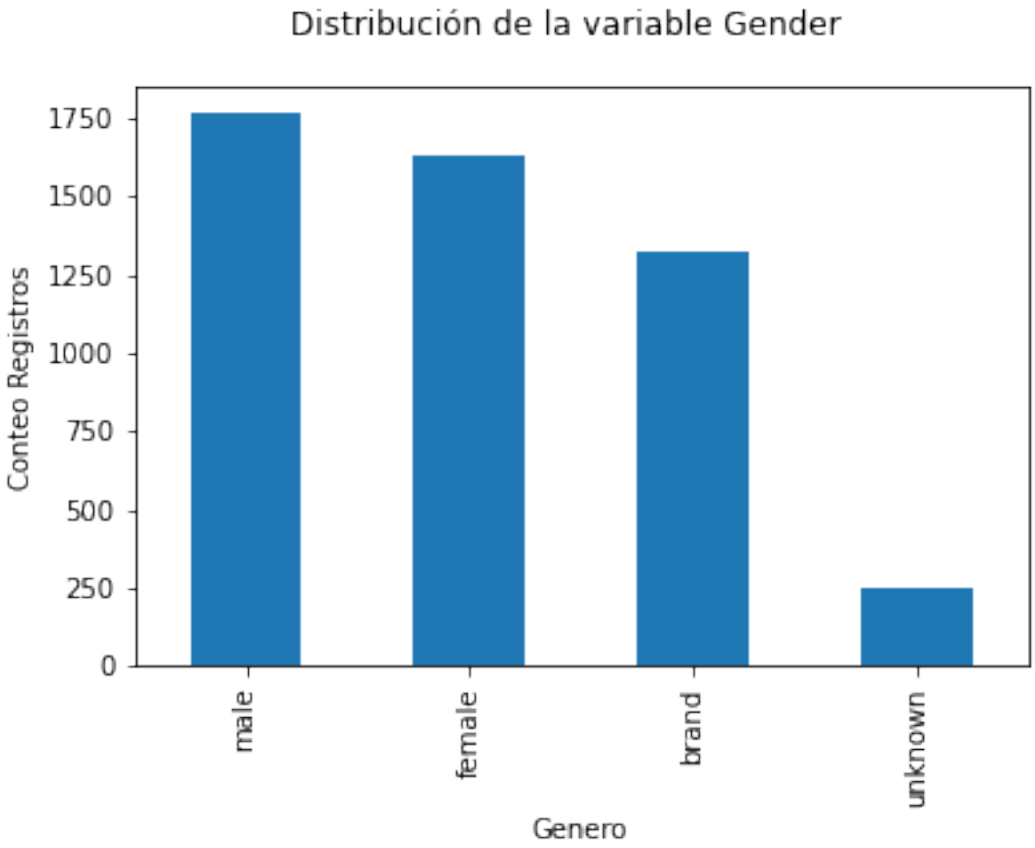
Univariado variables String y Bool

[9]:

	count	unique	\
_golden	5000	1	
_unit_state	5000	1	
_last_judgment_at	5000	281	
gender	4964	4	
profile_yn	5000	2	
created	5000	4764	
description	4329	4104	
link_color	5000	906	
name	5000	4771	
profileimage	5000	4635	
sidebar_color	5000	189	
text	5000	4895	
tweet_coord	53	47	
tweet_created	5000	1	
tweet_location	3338	2433	
user_timezone	3265	122	

	top	freq
_golden	False	5000
_unit_state	finalized	5000
_last_judgment_at	10/26/15 23:05	53
gender	male	1763
profile_yn	yes	4964
created	4/13/10 2:10	18
description	You can be spiritually empowered, financially …	33
link_color	0084B4	2277
name	naijama	18
profileimage	https://abs.twimg.com/sticky/default_profile_i…	29
sidebar_color	C0DEED	2083
text	Get Weather Updates from The Weather Channel. …	35
tweet_coord	[40.798598, -73.971836]	2
tweet_created	10/26/15 12:40	5000
tweet_location	United States	53
user_timezone	Eastern Time (US & Canada)	673

- Realice un análisis grafico que permita ver la distribución y ver las correlaciones

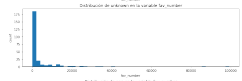


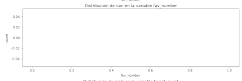

```
Relación de distribución de Genero con otras variables
--------------------------------------------------------------------------------
--------------------
numero de gráficos: 25
```

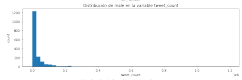

Distribución de male en la variable fav_number
count
fav_number


Distribución de female en la variable fav_number
count
fav_number

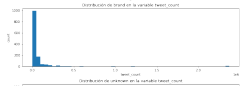

Distribución de brand en la variable fav_number
count
fav_number


Distribución de unknown en la variable fav_number
count
fav_number


Distribución de nan en la variable fav_number
count
fav_number


Distribución de male en la variable tweet_count
count
tweet_count


Distribución de female en la variable tweet_count
count
tweet_count

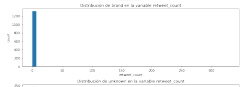

Distribución de brand en la variable tweet_count
count
tweet_count


Distribución de unknown en la variable tweet_count
count
tweet_count

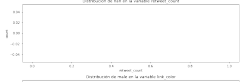

Distribución de nan en la variable tweet_count
count
tweet_count


Distribución de male en la variable retweet_count
count
retweet_count


Distribución de female en la variable retweet_count
count
retweet_count


Distribución de brand en la variable retweet_count
count
retweet_count

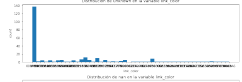

Distribución de unknown en la variable retweet_count
count
retweet_count

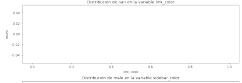

Distribución de nan en la variable retweet_count
count
retweet_count


Distribución de male en la variable link_color
count
link_color


Distribución de female en la variable link_color
count
link_color


Distribución de brand en la variable link_color
count
link_color


Distribución de unknown en la variable link_color
count
link_color

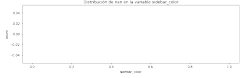

Distribución de nan en la variable link_color
count
link_color


Distribución de male en la variable sidebar_color
count
sidebar_color

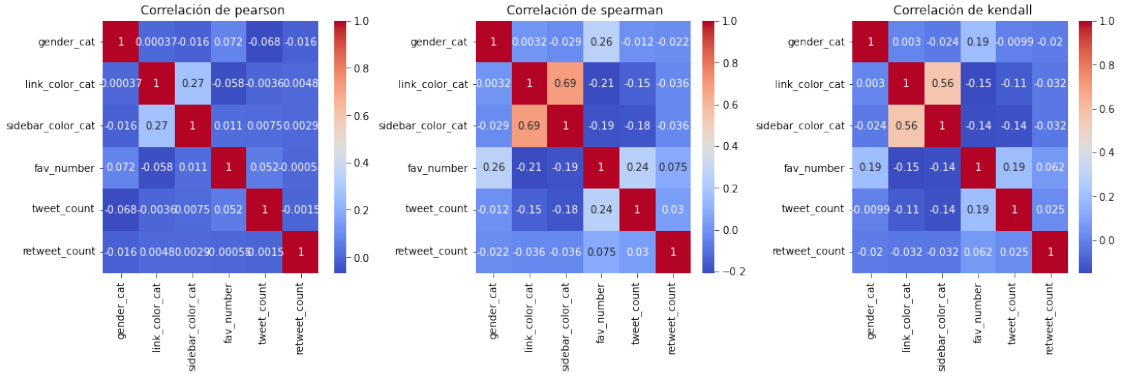

Distribución de female en la variable sidebar_color
count
sidebar_color


Distribución de brand en la variable sidebar_color
count
sidebar_color


Distribución de unknown en la variable sidebar_color
count
sidebar_color


Distribución de nan en la variable sidebar_color
count
sidebar_color

```
Relación de correlación
--------------------------------------------------------------------------------
--------------------
```



Comentarios

Despues de realizar un análisis de distribución y de correlación de las variables que consideramos de interes, podemos resaltar los siguientes casos:

- La variable respuesta requiere una limpieza previa con respecto a la categoría NaN que no representa aporte al ejercicio (Por fortuna es menos del 10% de la base).
- Dado que la correlación se hace entre variables categoricas no ordinales, la mejor correlación a estudiar es la de Kendall, la cual nos muestra la no correlación entre las variables. Solamente vemos una relación positiva alta para link_color y sidebar_color, pero con respecto a la variable de interes es baja. Lo cual es bueno, para evitar problemas de sobrepeso de una variable en el ejercicio de redes.
- La distribución de las covariables con respecto al genero, no varía, salvo en la cantidad de frecuencia, lo cual puede llevar a futuro a un bajo nivel de accuracy al momento de clasificar.

1.3 3. Limpieza de datos

De las variables que componen el problema cuales pueden ser implementadas para la construcción del modelo, justifique la respuesta.

```
(5000, 29)
(0, 29)

Variables seleccionadas y target
--------------------------------------------------------------------------------
--------------------
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4964 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype
```

```
---  ------         --------------  -----
 0   link_color     4964 non-null   object
 1   sidebar_color  4964 non-null   object
 2   fav_number     4964 non-null   int64
 3   retweet_count  4964 non-null   int64
 4   tweet_count    4964 non-null   int64
 5   gender         4964 non-null   object
dtypes: int64(3), object(3)
memory usage: 271.5+ KB
None
```

1.4 4. Utilización variables categóricas

Revisando las variables del problema realice una transformación de las variables categóricas para que puedan ser analizadas en el problema y volver a construir el modelo

Comentarios Despues de Limpiar y categorizar las variables, para la creación del modelo, usaremos las variables:

Target - gender_cat *Features* - fav_number - retweet_count - tweet_count - link_color_cat - sidebar_color_cat

1.5 5. Construcción del dataset

Para realizar el análisis de clasificación se sugiere realizar un particionamiento entre dos conjuntos (entrenamiento, pruebas) o tres conjuntos de entrenamiento, (entrenamiento, validación, pruebas) - ¿Qué diferencia hay en usar un conjunto de validación? - ¿mejora los resultados en la construcción del modelo usar un conjunto de validación? - ¿Qué información provee el uso de un conjunto de validación?

```
Training   X= (3474, 7910), y = (3474, 4)
Test   X= (745, 7910), y = (745, 4)
Validation   X= (745, 7910), y = (745, 4)
```

```
/home/jest/anaconda3/envs/javeriana_tf/lib/python3.9/site-
packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is
deprecated in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy
array with np.asarray. For more information see:
https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
/home/jest/anaconda3/envs/javeriana_tf/lib/python3.9/site-
packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is
deprecated in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy
array with np.asarray. For more information see:
https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
/home/jest/anaconda3/envs/javeriana_tf/lib/python3.9/site-
packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is
deprecated in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy
array with np.asarray. For more information see:
```

```
https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
/home/jest/anaconda3/envs/javeriana_tf/lib/python3.9/site-
packages/sklearn/utils/validation.py:593: FutureWarning: np.matrix usage is
deprecated in 1.0 and will raise a TypeError in 1.2. Please convert to a numpy
array with np.asarray. For more information see:
https://numpy.org/doc/stable/reference/generated/numpy.matrix.html
  warnings.warn(
```

Comentarios - El usar un conjunto de validación, es la prueba final de ajuste del modelo, mientras el proceso de training y test permiten crear el modelo y validarlo, durante la craci´on del mismo. La muestra de validación, permite un chequeo final, como si los datos entraran a producción, para tener los resultados finales del modelo propuesto. - Al usar una muestra de validación, permite al investigador realizar ajustes o modificaciones sobre el modelo, ya que es la muestra que tiene como evidencia de como se comporta el fenomeno de estudio, permitiendo modificar la arquitectura construida y así mejorar los resultados. - Evalua el comportamiento del modelo ya entrenado y validado con la muestra de test.

1.6 6. Elaboración del modelo

Diseñar diferentes soluciones empleando diferentes arquitecturas #### Perceptrón

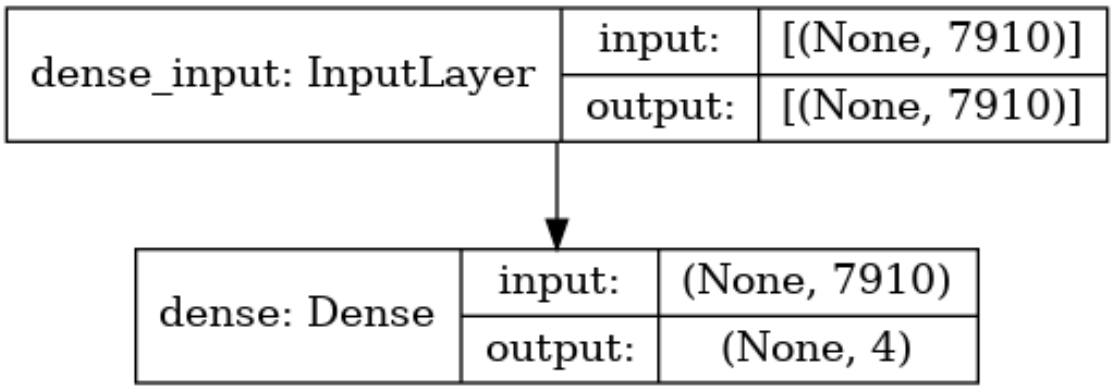
```
(4964, 7910)
```

```
2022-04-05 20:56:06.346215: I tensorflow/compiler/jit/xla_cpu_device.cc:41] Not
creating XLA devices, tf_xla_enable_xla_devices not set
2022-04-05 20:56:06.346838: W
tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load
dynamic library 'libcuda.so.1'; dlerror: libcuda.so.1: cannot open shared object
file: No such file or directory
2022-04-05 20:56:06.346865: W
tensorflow/stream_executor/cuda/cuda_driver.cc:326] failed call to cuInit:
UNKNOWN ERROR (303)
2022-04-05 20:56:06.346899: I
tensorflow/stream_executor/cuda/cuda_diagnostics.cc:156] kernel driver does not
appear to be running on this host (jest): /proc/driver/nvidia/version does not
exist
2022-04-05 20:56:06.347309: I tensorflow/core/platform/cpu_feature_guard.cc:142]
This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical
operations:  SSE4.1 SSE4.2 AVX AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2022-04-05 20:56:06.347742: I tensorflow/compiler/jit/xla_gpu_device.cc:99] Not
creating XLA devices, tf_xla_enable_xla_devices not set
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
```

```
dense (Dense)                (None, 4)                 31644
=================================================================
Total params: 31,644
Trainable params: 31,644
Non-trainable params: 0
_________________________________________________________________
```

[23]:




```
Epoch 1/30
218/218 - 0s - loss: 1.8443 - accuracy: 0.2775
Epoch 2/30
218/218 - 0s - loss: 0.2286 - accuracy: 0.9611
Epoch 3/30
218/218 - 0s - loss: 0.1320 - accuracy: 0.9729
Epoch 4/30
218/218 - 0s - loss: 0.1036 - accuracy: 0.9775
Epoch 5/30
218/218 - 0s - loss: 0.0868 - accuracy: 0.9813
Epoch 6/30
218/218 - 0s - loss: 0.0770 - accuracy: 0.9836
Epoch 7/30
218/218 - 0s - loss: 0.0688 - accuracy: 0.9839
Epoch 8/30
218/218 - 0s - loss: 0.0638 - accuracy: 0.9836
Epoch 9/30
218/218 - 0s - loss: 0.0593 - accuracy: 0.9850
Epoch 10/30
218/218 - 0s - loss: 0.0555 - accuracy: 0.9865
Epoch 11/30
218/218 - 0s - loss: 0.0513 - accuracy: 0.9876
Epoch 12/30
218/218 - 0s - loss: 0.0503 - accuracy: 0.9868
Epoch 13/30
218/218 - 0s - loss: 0.0480 - accuracy: 0.9876
Epoch 14/30
```

```
218/218 - 0s - loss: 0.0456 - accuracy: 0.9888
Epoch 15/30
218/218 - 0s - loss: 0.0436 - accuracy: 0.9893
Epoch 16/30
218/218 - 0s - loss: 0.0421 - accuracy: 0.9885
Epoch 17/30
218/218 - 0s - loss: 0.0402 - accuracy: 0.9908
Epoch 18/30
218/218 - 0s - loss: 0.0429 - accuracy: 0.9914
Epoch 19/30
218/218 - 0s - loss: 0.0414 - accuracy: 0.9905
Epoch 20/30
218/218 - 0s - loss: 0.0404 - accuracy: 0.9902
Epoch 21/30
218/218 - 0s - loss: 0.0381 - accuracy: 0.9902
Epoch 22/30
218/218 - 0s - loss: 0.0367 - accuracy: 0.9902
Epoch 23/30
218/218 - 0s - loss: 0.0348 - accuracy: 0.9917
Epoch 24/30
218/218 - 0s - loss: 0.0346 - accuracy: 0.9908
Epoch 25/30
218/218 - 0s - loss: 0.0340 - accuracy: 0.9893
Epoch 26/30
218/218 - 0s - loss: 0.0330 - accuracy: 0.9905
Epoch 27/30
218/218 - 0s - loss: 0.0329 - accuracy: 0.9914
Epoch 28/30
218/218 - 0s - loss: 0.0322 - accuracy: 0.9908
Epoch 29/30
218/218 - 0s - loss: 0.0310 - accuracy: 0.9914
Epoch 30/30
218/218 - 0s - loss: 0.0316 - accuracy: 0.9908
CPU times: user 12.3 s, sys: 1.47 s, total: 13.8 s
Wall time: 10.9 s
```

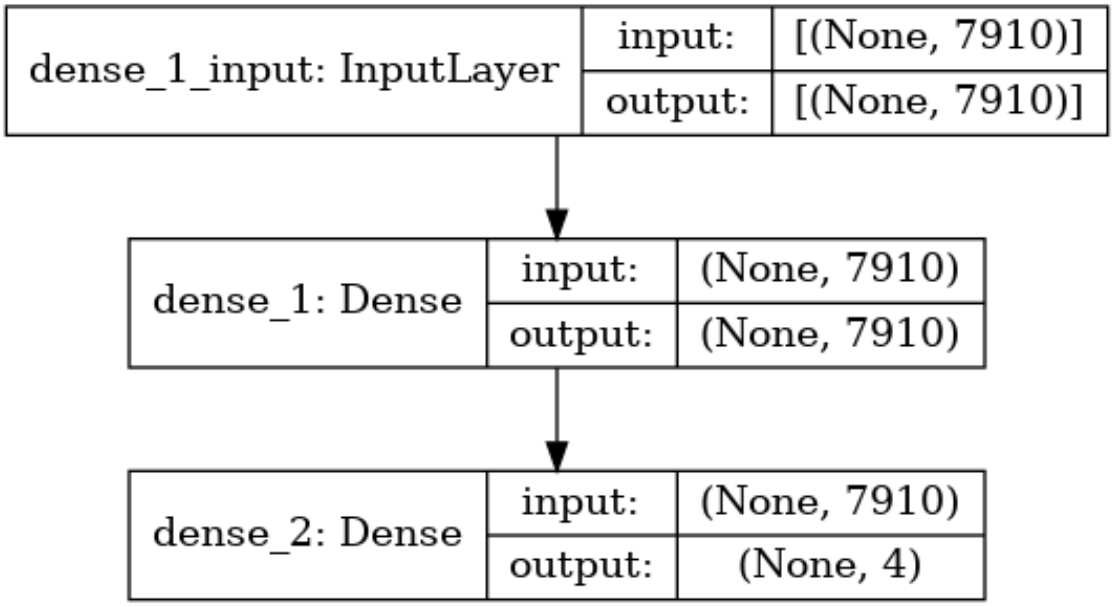
[29]:
```
<tensorflow.python.keras.callbacks.History at 0x7faf1c220e80>
```

```
24/24 - 0s - loss: 1.6685 - accuracy: 0.4443
```

Red neuronal con una capa oculta con un numero de neuronas igual al numero de entradas

```
(4964, 7910)
```

[25]:




```
Epoch 1/30
218/218 - 28s - loss: 1.3414 - accuracy: 0.3520
Epoch 2/30
218/218 - 29s - loss: 1.2899 - accuracy: 0.3520
Epoch 3/30
218/218 - 29s - loss: 1.2669 - accuracy: 0.3520
Epoch 4/30
218/218 - 28s - loss: 1.2553 - accuracy: 0.3520
Epoch 5/30
218/218 - 29s - loss: 1.2483 - accuracy: 0.3520
Epoch 6/30
218/218 - 29s - loss: 1.2440 - accuracy: 0.3520
Epoch 7/30
218/218 - 28s - loss: 1.2414 - accuracy: 0.3520
Epoch 8/30
218/218 - 28s - loss: 1.2396 - accuracy: 0.3520
Epoch 9/30
218/218 - 29s - loss: 1.2384 - accuracy: 0.3520
Epoch 10/30
218/218 - 29s - loss: 1.2376 - accuracy: 0.3520
Epoch 11/30
218/218 - 29s - loss: 1.2370 - accuracy: 0.3520
Epoch 12/30
218/218 - 28s - loss: 1.2366 - accuracy: 0.3520
Epoch 13/30
218/218 - 29s - loss: 1.2364 - accuracy: 0.3520
Epoch 14/30
```

```
218/218 - 29s - loss: 1.2361 - accuracy: 0.3520
Epoch 15/30
218/218 - 30s - loss: 1.2360 - accuracy: 0.3520
Epoch 16/30
218/218 - 29s - loss: 1.2358 - accuracy: 0.3520
Epoch 17/30
218/218 - 29s - loss: 1.2357 - accuracy: 0.3520
Epoch 18/30
218/218 - 29s - loss: 1.2357 - accuracy: 0.3520
Epoch 19/30
218/218 - 29s - loss: 1.2356 - accuracy: 0.3520
Epoch 20/30
218/218 - 29s - loss: 1.2356 - accuracy: 0.3520
Epoch 21/30
218/218 - 30s - loss: 1.2356 - accuracy: 0.3520
Epoch 22/30
218/218 - 29s - loss: 1.2355 - accuracy: 0.3520
Epoch 23/30
218/218 - 29s - loss: 1.2355 - accuracy: 0.3520
Epoch 24/30
218/218 - 29s - loss: 1.2355 - accuracy: 0.3520
Epoch 25/30
218/218 - 29s - loss: 1.2355 - accuracy: 0.3520
Epoch 26/30
218/218 - 29s - loss: 1.2355 - accuracy: 0.3520
Epoch 27/30
218/218 - 30s - loss: 1.2355 - accuracy: 0.3520
Epoch 28/30
218/218 - 29s - loss: 1.2355 - accuracy: 0.3520
Epoch 29/30
218/218 - 29s - loss: 1.2355 - accuracy: 0.3520
Epoch 30/30
218/218 - 29s - loss: 1.2355 - accuracy: 0.3520
CPU times: user 33min 7s, sys: 18min 41s, total: 51min 48s
Wall time: 14min 28s
```

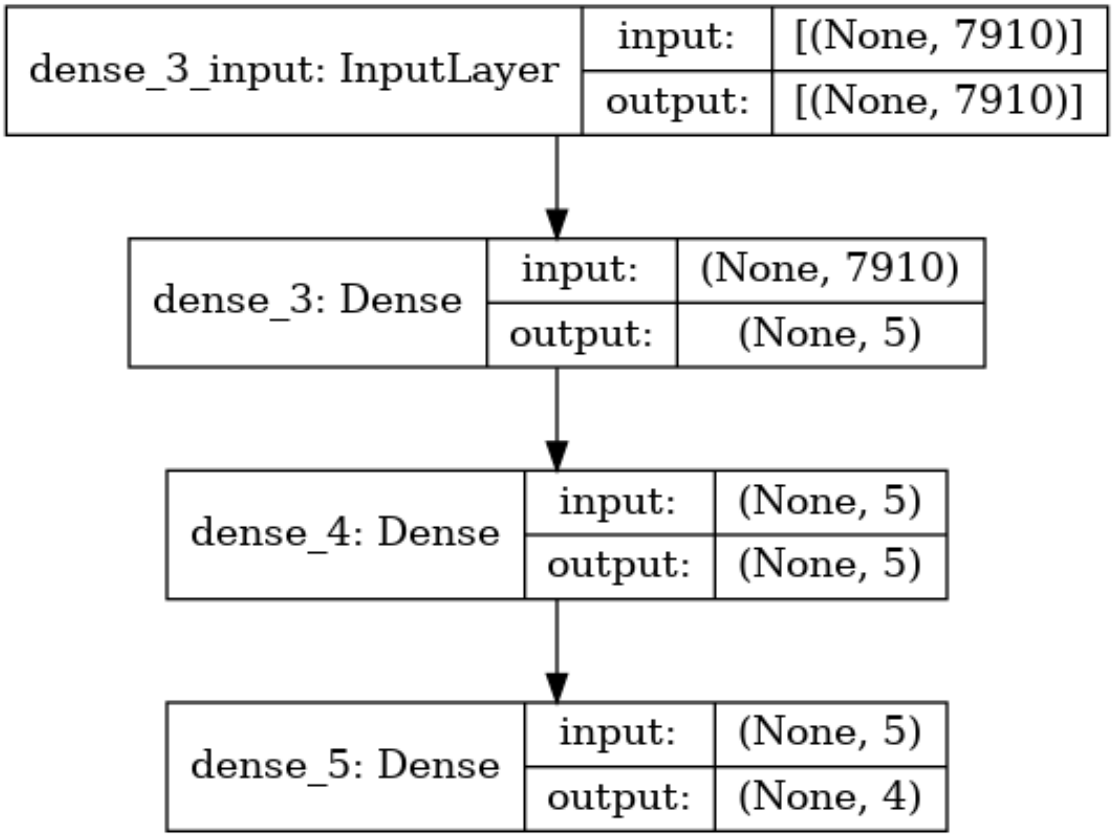
[33]:
```
<tensorflow.python.keras.callbacks.History at 0x7faeef88ec18>
```

```
24/24 - 1s - loss: 1.2405 - accuracy: 0.3544
```

Red neuronal con dos capas ocultas, la primera con 5 neuronas y la segunda capa oculta con 5 neuronas

```
(4964, 7910)
```

[27]:




```
Epoch 1/30
218/218 - 0s - loss: 1.3167 - accuracy: 0.3287
Epoch 2/30
218/218 - 0s - loss: 1.2476 - accuracy: 0.3710
Epoch 3/30
218/218 - 0s - loss: 1.2243 - accuracy: 0.4099
Epoch 4/30
218/218 - 0s - loss: 1.2092 - accuracy: 0.4188
Epoch 5/30
218/218 - 0s - loss: 1.1949 - accuracy: 0.4467
Epoch 6/30
218/218 - 0s - loss: 1.1792 - accuracy: 0.4810
Epoch 7/30
218/218 - 0s - loss: 1.1607 - accuracy: 0.5276
Epoch 8/30
218/218 - 0s - loss: 1.1384 - accuracy: 0.5656
Epoch 9/30
218/218 - 0s - loss: 1.1109 - accuracy: 0.6183
Epoch 10/30
218/218 - 0s - loss: 1.0777 - accuracy: 0.6926
Epoch 11/30
```

```
218/218 - 0s - loss: 1.0381 - accuracy: 0.7320
Epoch 12/30
218/218 - 0s - loss: 0.9923 - accuracy: 0.7893
Epoch 13/30
218/218 - 0s - loss: 0.9419 - accuracy: 0.8310
Epoch 14/30
218/218 - 0s - loss: 0.8887 - accuracy: 0.8497
Epoch 15/30
218/218 - 0s - loss: 0.8356 - accuracy: 0.8733
Epoch 16/30
218/218 - 0s - loss: 0.7845 - accuracy: 0.8903
Epoch 17/30
218/218 - 0s - loss: 0.7367 - accuracy: 0.9007
Epoch 18/30
218/218 - 0s - loss: 0.6924 - accuracy: 0.9105
Epoch 19/30
218/218 - 0s - loss: 0.6510 - accuracy: 0.9171
Epoch 20/30
218/218 - 0s - loss: 0.6123 - accuracy: 0.9229
Epoch 21/30
218/218 - 0s - loss: 0.5754 - accuracy: 0.9280
Epoch 22/30
218/218 - 0s - loss: 0.5398 - accuracy: 0.9341
Epoch 23/30
218/218 - 0s - loss: 0.5052 - accuracy: 0.9347
Epoch 24/30
218/218 - 0s - loss: 0.4718 - accuracy: 0.9361
Epoch 25/30
218/218 - 0s - loss: 0.4398 - accuracy: 0.9381
Epoch 26/30
218/218 - 0s - loss: 0.4093 - accuracy: 0.9398
Epoch 27/30
218/218 - 0s - loss: 0.3810 - accuracy: 0.9378
Epoch 28/30
218/218 - 0s - loss: 0.3550 - accuracy: 0.9401
Epoch 29/30
218/218 - 0s - loss: 0.3312 - accuracy: 0.9410
Epoch 30/30
218/218 - 0s - loss: 0.3101 - accuracy: 0.9398
CPU times: user 12.9 s, sys: 1.58 s, total: 14.4 s
Wall time: 8.54 s
```

[58]: <tensorflow.python.keras.callbacks.History at 0x7faf1e5546a0>

```
24/24 - 0s - loss: 1.3974 - accuracy: 0.4174
```

1.7 7. Análisis de Resultados

Construya la matriz de confusión, realice las metricas de evaluación (acuraccy, precisión, recall, f1 score) para cada uno de los modelos. Realice un análisis comparativo de cada uno de los resultados de cada modelo.

```
--------------------------------------------------------------------------------
--------------------
Perceptron
--------------------------------------------------------------------------------
--------------------
Matriz de confusión
--------------------------------------------------------------------------------
--------------------
[[123  19  45   3]
 [134  65  48   4]
 [155  36  66   7]
 [ 19  10   8   3]]
--------------------------------------------------------------------------------
--------------------
Evaluación del modelo
--------------------------------------------------------------------------------
--------------------
109/109 [==============================] - 0s 2ms/step - loss: 0.0253 -
accuracy: 0.9937
TRAINING
accuracy: 99.37%
24/24 [==============================] - 0s 2ms/step - loss: 1.6685 - accuracy:
0.4443
TESTING
accuracy: 44.43%
--------------------------------------------------------------------------------
--------------------
Metricas de Evaluación del modelo
--------------------------------------------------------------------------------
--------------------
              precision    recall  f1-score   support

           0       0.39      0.44      0.41       167
           1       0.26      0.50      0.34       130
           2       0.25      0.40      0.31       167
           3       0.07      0.18      0.11        17

   micro avg       0.28      0.43      0.34       481
   macro avg       0.24      0.38      0.29       481
weighted avg       0.29      0.43      0.35       481
 samples avg       0.28      0.28      0.28       481
```

```
--------------------------------------------------------------------------------
--------------------
1 capa densa mismas neuronas que features
--------------------------------------------------------------------------------
--------------------

/home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

Matriz de confusión
--------------------------------------------------------------------------------
--------------------
[[190   0   0   0]
 [251   0   0   0]
 [264   0   0   0]
 [ 40   0   0   0]]
--------------------------------------------------------------------------------
--------------------
Evaluación del modelo
--------------------------------------------------------------------------------
--------------------
109/109 [==============================] - 6s 54ms/step - loss: 1.2353 -
accuracy: 0.3520
TRAINING
accuracy: 35.20%
24/24 [==============================] - 1s 45ms/step - loss: 1.2405 - accuracy:
0.3544
TESTING
accuracy: 35.44%
--------------------------------------------------------------------------------
--------------------
Metricas de Evaluación del modelo
--------------------------------------------------------------------------------
--------------------
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       0.00      0.00      0.00         0
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0

   micro avg       0.00      0.00      0.00         0
   macro avg       0.00      0.00      0.00         0
weighted avg       0.00      0.00      0.00         0
 samples avg       0.00      0.00      0.00         0
```

```
--------------------------------------------------------------------------------
--------------------
dos capas densas de 5 neuronas cada una
--------------------------------------------------------------------------------
--------------------
Matriz de confusión
--------------------------------------------------------------------------------
--------------------
[[150  11  29   0]
 [159  53  39   0]
 [193  29  42   0]
 [ 26   9   5   0]]
--------------------------------------------------------------------------------
--------------------
Evaluación del modelo
--------------------------------------------------------------------------------
--------------------

/home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 due to no true samples. Use
`zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))

109/109 [==============================] - 0s 1ms/step - loss: 0.4839 -
accuracy: 0.9338
TRAINING
accuracy: 93.38%
24/24 [==============================] - 0s 990us/step - loss: 1.3824 -
accuracy: 0.3664
TESTING
accuracy: 36.64%
--------------------------------------------------------------------------------
--------------------
Metricas de Evaluación del modelo
--------------------------------------------------------------------------------
--------------------
              precision    recall  f1-score   support
```

```
              0       0.13      0.44      0.20        57
              1       0.21      0.52      0.30       102
              2       0.16      0.37      0.22       115
              3       0.00      0.00      0.00         0

   micro avg       0.16      0.44      0.24       274
   macro avg       0.13      0.33      0.18       274
weighted avg       0.17      0.44      0.25       274
 samples avg       0.16      0.16      0.16       274


/home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in labels with no true samples.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
/home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-
packages/sklearn/metrics/_classification.py:1245: UndefinedMetricWarning: Recall
and F-score are ill-defined and being set to 0.0 in samples with no true labels.
Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

Comentarios

Despues de crear los modelos propuestos (Para este caso usamos una función de activación única softmax (Para el modelo 3 cambiamos a funciones sigmoid en las capas ocultas), un optimizador basado en gradiente descendiente estocastico y una función de perdida de cossentropy categorica) y para medir la eficiencia del modelo, usamos el accuracy.

resaltamos lo siguiente:

- En tiempos de ejecución el más complejo es el modelo con dos capas con el mismo número de neuronas a datos de entrada. Y el más rápido el perceptron.
- Curiosamente, para el modelo 2, su accuracy fue del 35% para la muestra de entrenamiento, siendo el mas bajo, mientras que el perceptron, tuvo un accuracy de entrenamiento del 99.37%.
- El modelo de perceptron tuvo un accuracy mayor con los datos de testing con un 44.4% de accuracy, seguido del modelo 3 con un accuracy del 41.7%.
- El perceptron tuvo una precisión más homogenia con respecto a los dos modelos siguientes. En el caso del modelo 3, la precisión solo no causo efecto en una categoria.
- Observo adicional, que el modelo 3 tiene probabilidad para competir con el modelo 1, si tuvieramos una variabilidad de neuronas, para mejorar el aprendizaje.

Para estas condiciones y la muestra construida, el modelo elegido sería el perceptron, pero en todos los casos con la muestra de test, su precisión baja a un 40% lo cual significa que le modelo puede mejorar.

1.8 8. Ajustes

- Explique que otros ajustes puede hacerse al dataset, modelo o técnica para construir un nuevo modelo con el cual posiblemente se pueda mejorar la predicción y modelamiento de los datos.
- Considere cambios de la función de activación, aumento o disminución de neuronas y/o capas, funcion de aprendizaje
- Tratamiento adicional de los datos
- Rendimiento, ¿es necesario agregar mas neuronas o capas?, métodos de entrenamiento (secuencial, batch)

El plantear la hipotesis no necesariamente considera su desarrollo, simplemente justificar las modificaciones que realizaría para mejorar los resultados y obtener un mejor modelo.

Solución

Las respuestas de este punto las hago en torno al modelo 3. 1) Para mejorar el modelo 3, podriamos hacer una validación múltiple entre funciones de activación para las capas ocultas, cambio de optimizador y de función de perdida. Ya con ello observar, cuales son las mejoras que se tengan en el cambio. Adicionalmente. Tener una validación final con la muestra de validación que no usamos, para ver la mejora dentro del modelo.

2) Tener un número grande de neuronas para este caso no genero una solución apropiada al modelo, pero depronto si hacemos una variación pequeña entre número de neuronas o activación, podriamos tener mejores resultados.

3) Podriamos adicionar información de texto, con el fín de tener mayores caracteristicas que nos permitan decidir si el twitter fue escrito por un genero particular, esto puede facilitar los procesos de análisis.

4) En terminos de rendimiento del modelo. El agregar más neuronas o cambio de proceso, puede afectar el tema de aprendizaje, incrementa la capacidad computacional, pero en los procesos iterativos, puede apoyarnos a encontrar un mejor performance en la clasificación del modelo.

1.9 9. Bono

- En base a las hipótesis planteadas en el punto anterior realizar la construcción del modelo.
- ¿mejoro o empeoró el modelo?, ¿la hipótesis fue correcta?, justifique su respuesta

```
/home/ec2-user/anaconda3/envs/amazonei_tensorflow2_p36/lib/python3.6/site-
packages/joblib/externals/loky/process_executor.py:691: UserWarning: A worker
stopped while some jobs were given to the executor. This can be caused by a too
short worker timeout or by a memory leak.
  "timeout or by a memory leak.", UserWarning
```

```
CPU times: user 6min 8s, sys: 9min 16s, total: 15min 24s
Wall time: 30min 10s
```

```
Best: 0.402994 using {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
```

```
0.362119 (0.030157) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
```

```
'adam'}
0.360679 (0.008172) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.385435 (0.030785) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.346575 (0.017059) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.365285 (0.017627) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.359816 (0.013329) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.318941 (0.064223) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.354634 (0.018009) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.361255 (0.037112) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.371906 (0.016344) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.375360 (0.019997) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.350892 (0.009469) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.356937 (0.040174) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.376223 (0.008816) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.293322 (0.029448) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
```

```
'adam'}
0.362119 (0.007063) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.321531 (0.059288) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.370754 (0.012014) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.338515 (0.091692) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.370466 (0.013599) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.303397 (0.052530) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.356074 (0.018310) with: {'activation_functions': 'relu', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.354634 (0.002850) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.341969 (0.014857) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.395222 (0.008471) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.349741 (0.005771) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.335636 (0.054513) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.347150 (0.002443) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.351180 (0.016344) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.339666 (0.010395) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.337939 (0.044467) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
```

```
'adam'}
0.363558 (0.003731) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.350029 (0.006359) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.357225 (0.013235) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.324410 (0.026722) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.360679 (0.008616) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.365861 (0.042025) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.368451 (0.005997) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.309729 (0.028748) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.360967 (0.006726) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.339666 (0.011398) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.356362 (0.018845) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.376799 (0.021110) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.340242 (0.006726) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.332182 (0.024530) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.347150 (0.011650) with: {'activation_functions': 'relu', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.349165 (0.014178) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
```

```
'adam'}
0.347150 (0.018128) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.393207 (0.003332) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.333333 (0.004624) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.364997 (0.036809) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.331031 (0.004799) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.367012 (0.009220) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.277490 (0.018779) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.349741 (0.010855) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.343408 (0.052369) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.356362 (0.006160) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.349165 (0.017476) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.330743 (0.018695) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.324410 (0.045331) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.355210 (0.022840) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.349453 (0.052340) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
```

```
'adam'}
0.341969 (0.016581) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.335348 (0.042647) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.335924 (0.037529) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.360967 (0.012233) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.332182 (0.039575) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.353483 (0.014490) with: {'activation_functions': 'relu', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.369891 (0.030387) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.355210 (0.017646) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.396373 (0.012867) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.352907 (0.018350) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.376223 (0.025131) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
```

```
'adam'}
0.360967 (0.012213) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.399252 (0.008111) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.354059 (0.019743) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.345423 (0.030928) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.340530 (0.011398) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.395509 (0.011861) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.354634 (0.019759) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.388601 (0.009538) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.360391 (0.021460) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.369027 (0.015356) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.353771 (0.017505) with: {'activation_functions': 'softmax', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.322107 (0.005596) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
```

```
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.354059 (0.011543) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.354922 (0.016581) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.372193 (0.016217) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.338803 (0.031630) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.352332 (0.015892) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.307714 (0.064227) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.353195 (0.017796) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.370754 (0.004695) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.360679 (0.022589) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.337651 (0.045679) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
```

```
'adam'}
0.351180 (0.015324) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.351756 (0.042343) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.355498 (0.003179) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.380829 (0.013618) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.322107 (0.005596) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.338803 (0.015866) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.335924 (0.010855) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.329879 (0.003525) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.349741 (0.005507) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.344560 (0.019553) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.365285 (0.014672) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
```

```
'adam'}
0.353195 (0.018695) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.387162 (0.015469) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.348877 (0.017542) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.344847 (0.040112) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.345135 (0.015210) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.383420 (0.022196) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.352332 (0.017670) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.352044 (0.041752) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.345999 (0.010794) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.372193 (0.012233) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.352907 (0.017476) with: {'activation_functions': 'softmax', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.311744 (0.040511) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
```

```
'adam'}
0.337939 (0.011398) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.370754 (0.044624) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.345999 (0.017898) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.392055 (0.024466) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.339954 (0.008263) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.318077 (0.016084) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.357801 (0.002850) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.364421 (0.038510) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.373345 (0.030526) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.400403 (0.029717) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.370466 (0.009166) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.350604 (0.030605) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.358377 (0.011281) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.385435 (0.025081) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.369315 (0.015485) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.270581 (0.045051) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
```

```
'adam'}
0.371042 (0.030133) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.333621 (0.028696) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.354922 (0.020809) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.298503 (0.064169) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.362406 (0.015866) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.330455 (0.044573) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.401554 (0.023692) with: {'activation_functions': 'sigmoid', 'batch_size': 10,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.363270 (0.007406) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.340242 (0.023053) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.333045 (0.063035) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.344847 (0.014829) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.374208 (0.038441) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.346862 (0.013103) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.349741 (0.015007) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.345135 (0.011464) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.387162 (0.024121) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
```

```
'adam'}
0.344272 (0.014178) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.390616 (0.013235) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.350317 (0.005292) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.342832 (0.011193) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.351756 (0.015597) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.386586 (0.005743) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.344560 (0.009247) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.402994 (0.014796) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.366724 (0.007539) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.341969 (0.035064) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.354059 (0.018237) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.356074 (0.019595) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.365285 (0.003664) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.344272 (0.078308) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.373345 (0.001774) with: {'activation_functions': 'sigmoid', 'batch_size': 20,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.352044 (0.017333) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
```

```
'adam'}
0.348877 (0.021926) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.361831 (0.020104) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.344272 (0.010770) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.371042 (0.040445) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.341969 (0.012233) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 5, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.333621 (0.015389) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.352044 (0.017333) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.343696 (0.035668) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.343408 (0.009066) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.372193 (0.020178) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.345423 (0.016966) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 10, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.336500 (0.007063) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.350604 (0.015703) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.373057 (0.025501) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.352044 (0.015677) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.367012 (0.044527) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
```

```
'adam'}
0.343984 (0.006701) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 15, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
0.308578 (0.023218) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'adam'}
0.333333 (0.008807) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 1, 'optimizer_':
'sgd'}
0.369027 (0.044802) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'adam'}
0.352044 (0.012138) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 5, 'optimizer_':
'sgd'}
0.392055 (0.000705) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'adam'}
0.337363 (0.010079) with: {'activation_functions': 'sigmoid', 'batch_size': 40,
'epochs': 20, 'loss_': 'categorical_crossentropy', 'neurons': 10, 'optimizer_':
'sgd'}
```

Comentarios

Al realizar las variaciones del modelo, logramos encontrar una variación de los procesos de epocas y batch, en ellos encontramos una estabilidad del modelo con un número menor de epocas y un manejo apropiado del tamaño de batch para la información.

Un dato adicional, es que el número de neuronas que permite un desarrollo apropiado del modelo es de 10 neuronas, para incrementar el accuracy a un 40%.