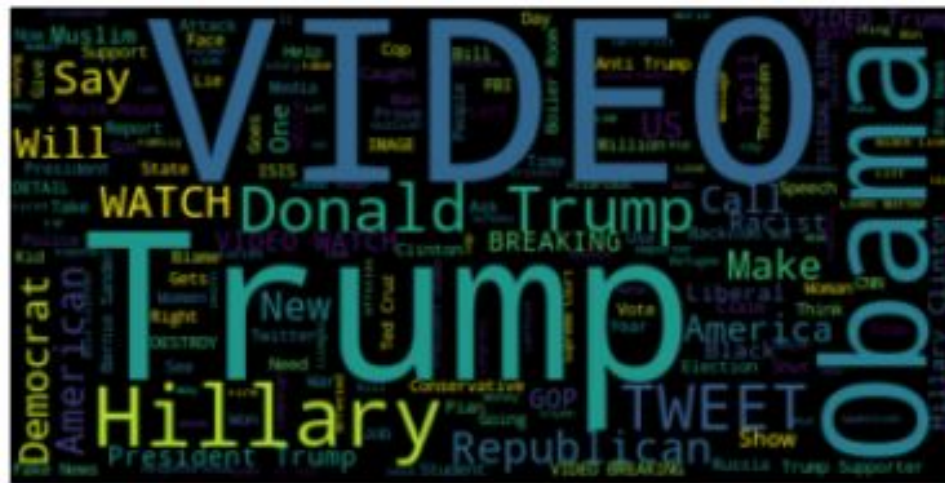


- NN para detectar Fake news
- Utilizamos dos datasets que contienen el texto de noticias a los que les creamos una clase para las reales (0) y las falsas (1)
- Visualizamos las palabras más frecuentes en cada clase

FAKE



TRUE



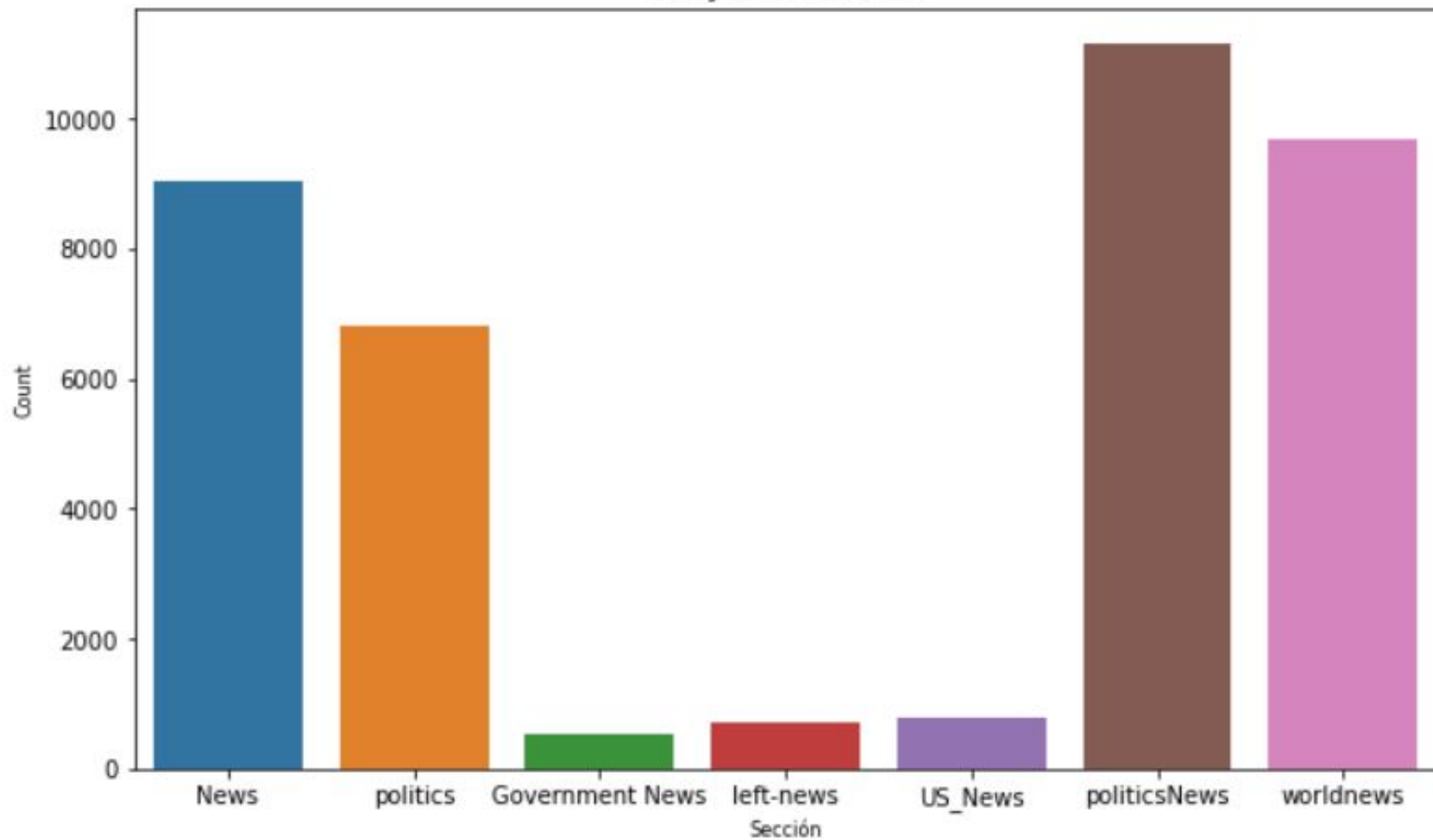
-En la nube de palabras de noticias verdaderas aparecen palabras que indican condicional.

Podría interpretarse como que en noticias falsas no es relevante la rigurosidad de que algo esté confirmado o no.

-Aparecen de forma más notoria palabras como WATCH y VIDEO en la nube de fake news

-En las noticias verdaderas hay palabras como OFFICIAL o SOURCE y en las fake, no

Subject Noticias



-Detectamos duplicados, unimos título y texto de la noticia en una misma variable

-A los subjects que figuran con diferente nombre en cada dataset los unimos a una misma categoría para que no queden subjects exclusivos de cada clase

```
[27]: #unificamos categorías similares

subject_replace = {
    "politicsNews" : "politics",
    "US_News" : "News",
    "Government News" : "politics",
    "left-news" : "politics"
}

df["new_subject"] = df["subject"]

for key in subject_replace:
    df["new_subject"] = df.new_subject.str.replace(key, subject_replace[key])

df.new_subject.value_counts(1)
```

```
[27]: politics      0.493338
worldnews      0.255210
News           0.251451
Name: new_subject, dtype: float64
```

-Generamos más variables que extraemos de la fecha

new_subject_News	new_subject_politics	new_subject_worldnews	year	news	month_Apr	month_Aug	month_Dec	month_Feb	month_Jan	month_Jul
1	0	0	2017	Donald Trump Sends Out Embarrassing New Year'...	0	0	1	0	0	0
1	0	0	2017	Drunk Bragging Trump Staffer Started Russian ...	0	0	1	0	0	0
				Sheriff David Clarke						

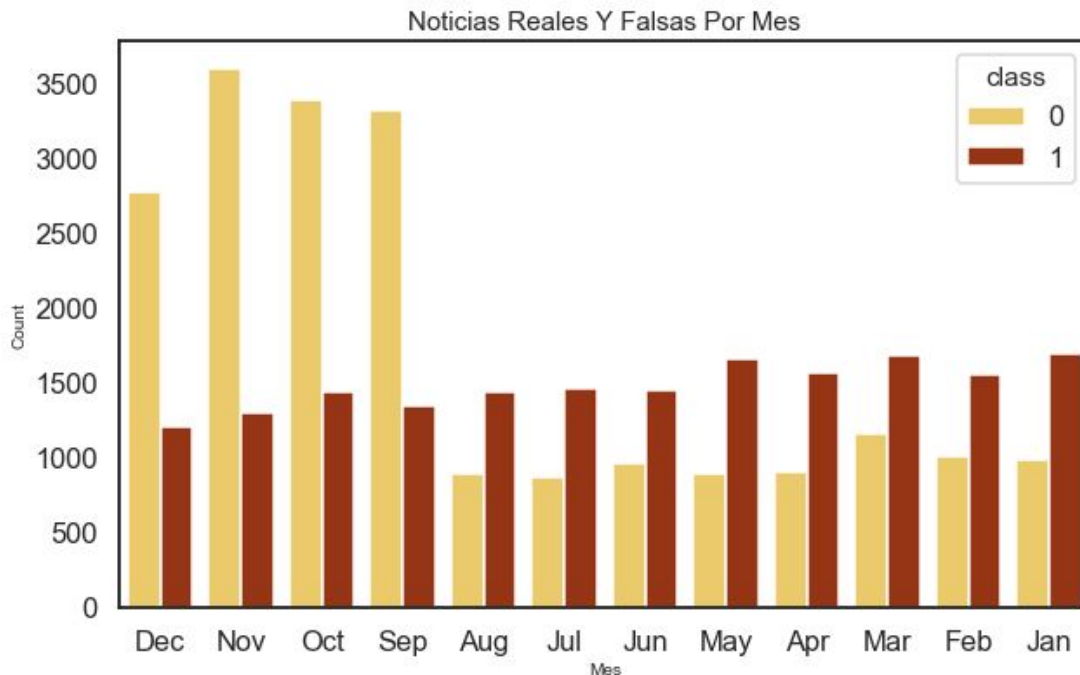
Nos fijamos la cantidad de datos que hay para cada mes y año

```
[31]: #Extraemos los años
      pattern_year=r'(\d{4})'
      df['year']=df.date.str.extract(pattern_year)
      df.year.value_counts()
```

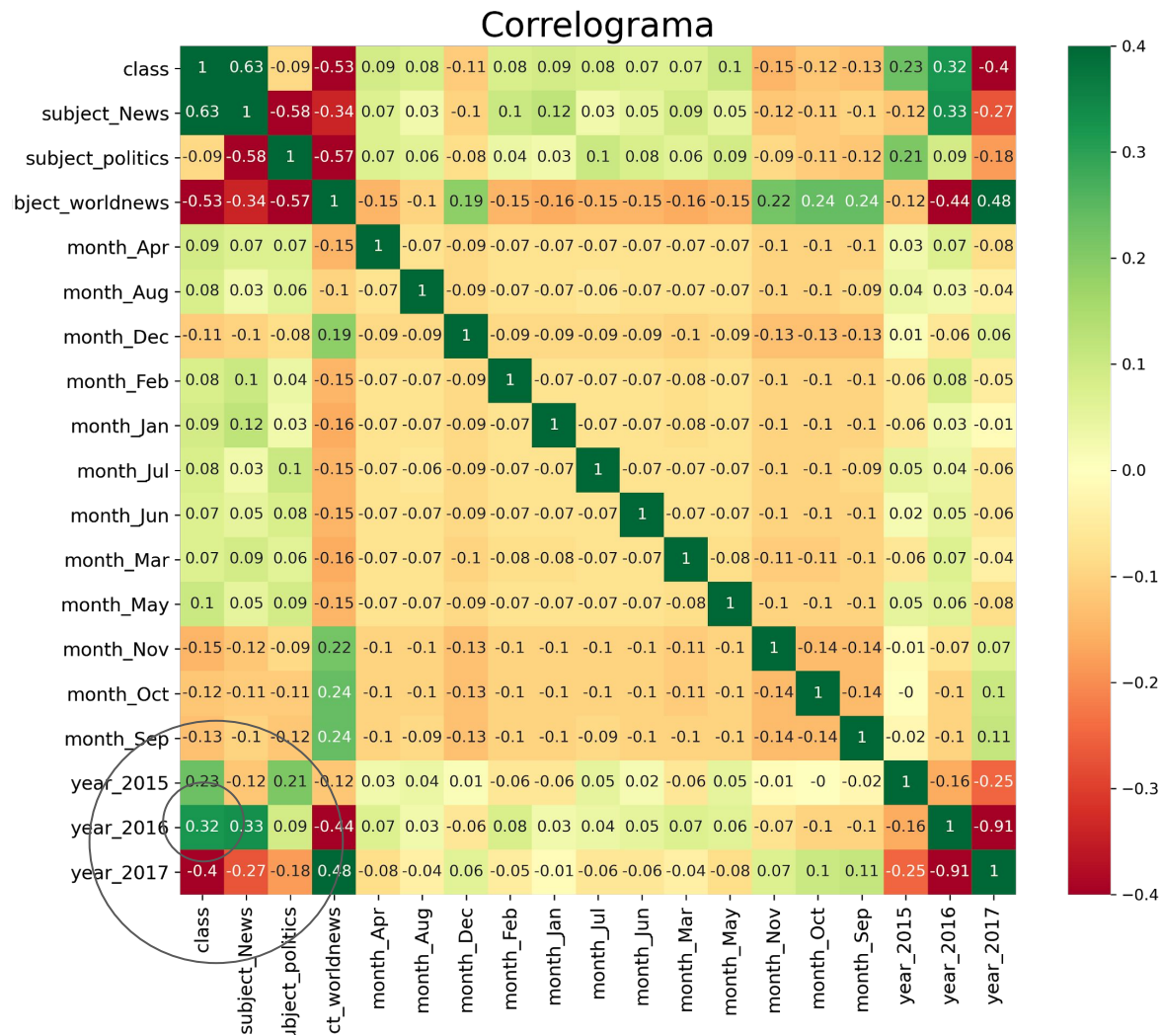
```
[31]: 2017    22951
      2016    14093
      2015     1646
      Name: year, dtype: int64
```

2017 es el año con más noticias pero no el año con más fake news

Mayo, Marzo y Enero son los meses con más fake news. Las elecciones son en Noviembre



2016 es el año con más fake news. Tiene sentido porque fue el año de las elecciones



Bag of words/CountVectorizer

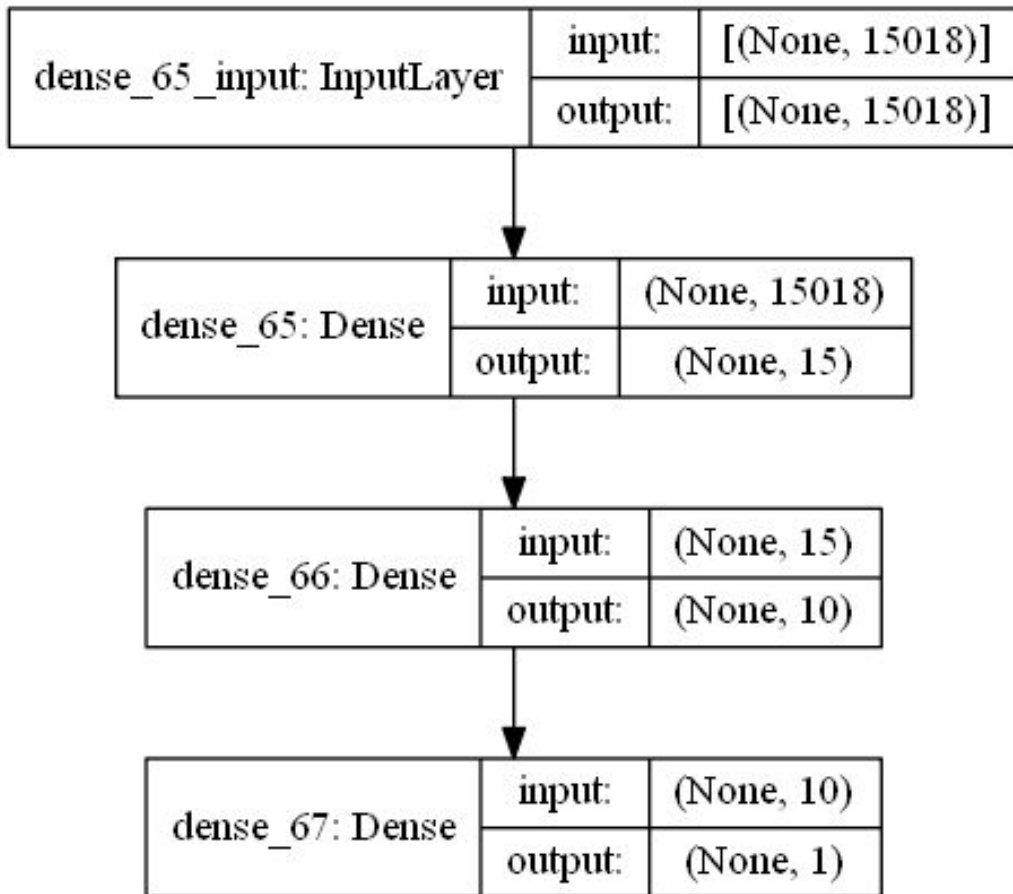
- Iteramos sobre cada oración para pasar el texto de las noticias a minúscula, remover puntos y espacios
- Tokenizamos las oraciones y creamos un diccionario que contiene la frecuencia con la que aparecen las palabras
- Transformamos las oraciones a una representación vectorial para que aparezca 1 si esa palabra está en la oración y 0 si no está
- Probamos hacer esto de forma manual y finalmente usamos CountVectorizer porque tardaba menos. Agregamos stopwords

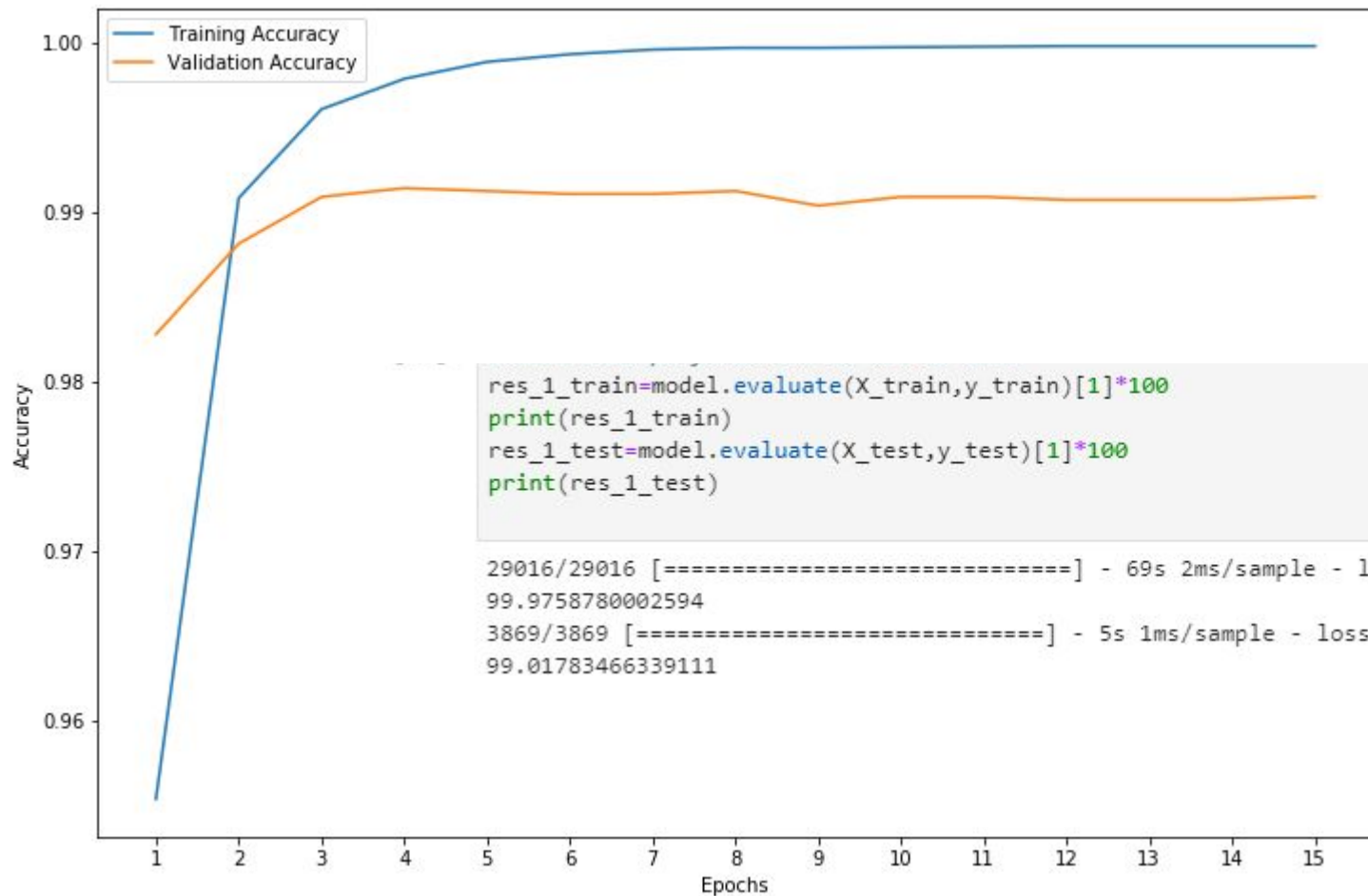
<https://stackabuse.com/python-for-nlp-creating-bag-of-words-model-from-scratch/>

Cosas que fuimos probando:

- Pasarle a la Red Neuronal sólo las palabras vectorizadas para que clasifique
- Pasarle las palabras vectorizadas + el resto de las variables (meses, subject) y comparar si había diferencia
- Red neuronal con regularización L1, L2, dropout rate (sin regularizar overfitteaba)
- La misma red neuronal con 1 y 2 capas. Nos quedamos con la de una capa
- Sumamos KerasClassifier, EarlyStopping, ReduceLROnPlateau, RandomizedSearch y cross validamos con 3 folds

Red Neuronal sin callbacks ni regularización





```
res_1_train=model.evaluate(X_train,y_train)[1]*100
print(res_1_train)
res_1_test=model.evaluate(X_test,y_test)[1]*100
print(res_1_test)
```

```
29016/29016 [=====] - 69s 2ms/sample - loss: 0.0040 - acc: 0.9998
99.9758780002594
3869/3869 [=====] - 5s 1ms/sample - loss: 0.0440 - acc: 0.9902
99.01783466339111
```

dense_3_input: InputLayer	input:	[(None, 15018)]
	output:	[(None, 15018)]



dense_3: Dense	input:	(None, 15018)
	output:	(None, 15)



dropout: Dropout	input:	(None, 15)
	output:	(None, 15)

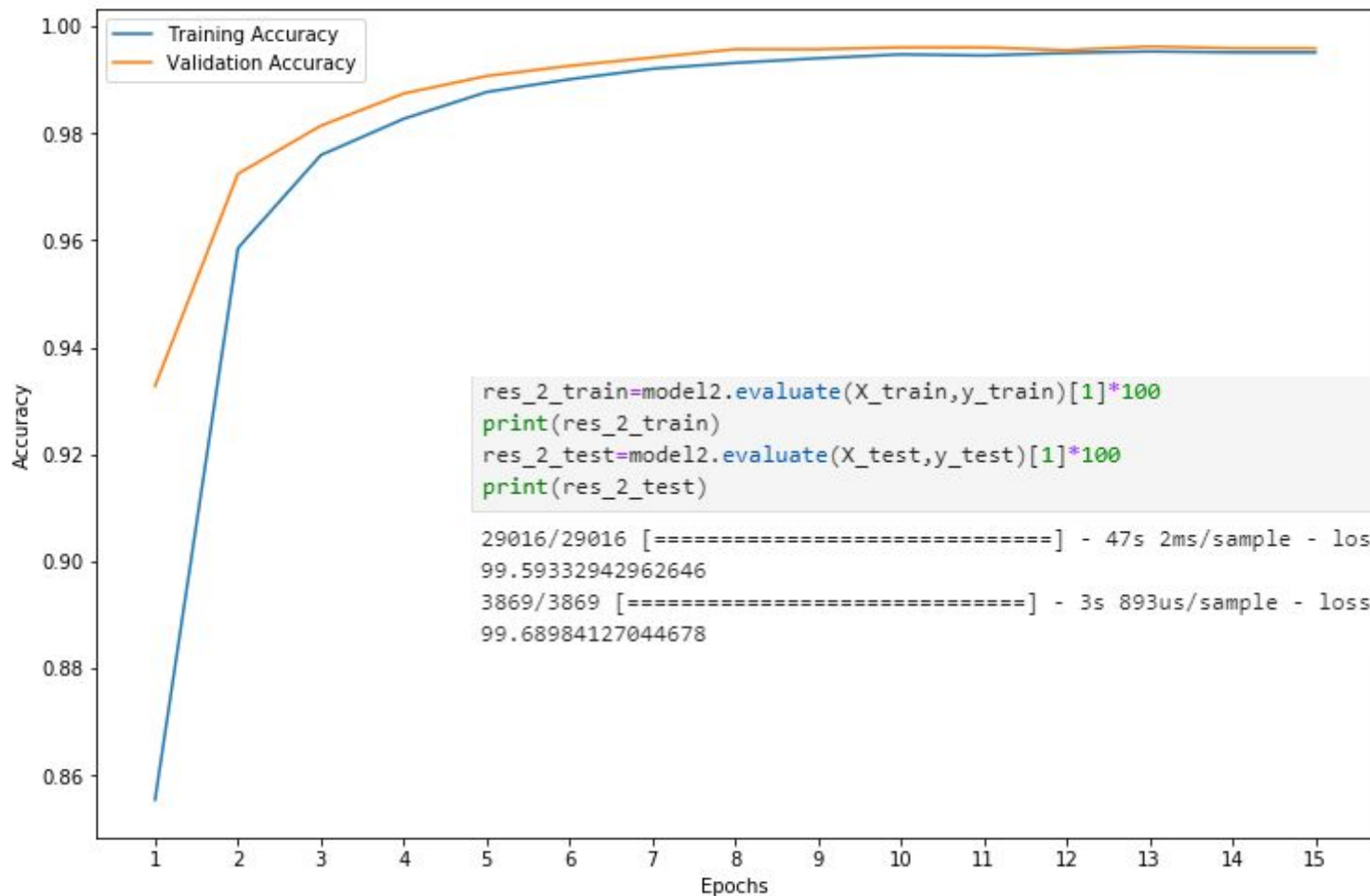


dense_4: Dense	input:	(None, 15)
	output:	(None, 10)



dense_5: Dense	input:	(None, 10)
	output:	(None, 1)

Red Neuronal con regularización y Dropout



```
res_2_train=model2.evaluate(X_train,y_train)[1]*100  
print(res_2_train)  
res_2_test=model2.evaluate(X_test,y_test)[1]*100  
print(res_2_test)
```

```
29016/29016 [=====] - 47s 2ms/sample - loss: 0.2034 - acc: 0.9959  
99.59332942962646  
3869/3869 [=====] - 3s 893us/sample - loss: 0.2014 - acc: 0.9969  
99.68984127044678
```

[85]:

	Model	Train Acc	Test Acc	Train Acc - Test Acc
1	Modelo 1	99.969	99.1212	0.847763
2	Modelo 2	99.5726	99.6123	-0.039655
3	Modelo 3	0.995726	0.996898	-0.00117193

Con Keras Classifier,
Early Stopping,
ReduceLROnPlateau,
Cross validation y
probando diferentes
dropout rates y units