

```
from google.colab import drive
drive.mount('/content/drive')
```

Mounted at /content/drive

```
import pandas as pd
import numpy as np
from keras import utils
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding
from keras.optimizers import RMSprop
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import to_categorical
%matplotlib inline

df = pd.read_csv('/content/drive/MyDrive/spam.csv', delimiter=',', encoding='latin-
df
```

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed: 4
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	NaN
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	NaN

```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
df
```

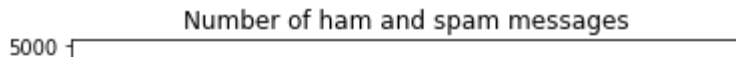
	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will i_b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...
5570	ham	The guy did some bitching but I acted like i'd...
5571	ham	Rofl. Its true to its name

5572 rows × 2 columns

```
sns.countplot(df.v1,palette='Set3')
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass t
FutureWarning
```

```
Text(0.5, 1.0, 'Number of ham and spam messages')
```



```
X = df.v2
```

```
Y = df.v1
```

```
le = LabelEncoder()
```

```
Y = le.fit_transform(Y)
```

```
Y = Y.reshape(-1,1)
```



```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```



```
max_words = 1000
```

```
max_len = 150
```

```
tok = Tokenizer(num_words=max_words)
```

```
tok.fit_on_texts(X_train)
```

```
sequences = tok.texts_to_sequences(X_train)
```

```
sequences_matrix = utils.pad_sequences(sequences,maxlen=max_len)
```

```
sequences_matrix.shape
```

```
(4736, 150)
```

```
sequences_matrix.ndim
```

```
2
```

```
sequences_matrix = np.reshape(sequences_matrix,(4736,150,1))
```

```
sequences_matrix.ndim
```

```
3
```

```
from keras.models import Sequential
```

```
from keras.layers import Dense
```

```
from keras.layers import LSTM
```

```
from keras.layers import Embedding
```

```
model = Sequential()
```

```
model.add(Embedding(max_words,50,input_length=max_len))
```

```
model.add(LSTM(units=64,input_shape = (sequences_matrix.shape[1],1),return_sequences=True))
```

```
model.add(LSTM(units=64,return_sequences=True))
```

```
model.add(LSTM(units=64,return_sequences=True))
```

```
model.add(LSTM(units=64))
```

```

model.add(Dense(units = 256,activation = 'relu'))
model.add(Dense(units = 1,activation = 'sigmoid'))

del.summary()
del.compile(loss='binary_crossentropy',optimizer='adam',metrics=['accuracy'])

```

Model: "sequential"

Layer (type)	Output Shape	Param #
embedding (Embedding)	(None, 150, 50)	50000
lstm (LSTM)	(None, 150, 64)	29440
lstm_1 (LSTM)	(None, 150, 64)	33024
lstm_2 (LSTM)	(None, 150, 64)	33024
lstm_3 (LSTM)	(None, 64)	33024
dense (Dense)	(None, 256)	16640
dense_1 (Dense)	(None, 1)	257
Total params: 195,409		
Trainable params: 195,409		
Non-trainable params: 0		

```

X = model.fit(sequences_matrix,Y_train,batch_size=128,epochs=5,validation_split=0.2)
X

```

```

Epoch 1/5
30/30 [=====] - 43s 1s/step - loss: 0.4513 - accuracy: 0.8461
Epoch 2/5
30/30 [=====] - 31s 1s/step - loss: 0.3183 - accuracy: 0.8870
Epoch 3/5
30/30 [=====] - 32s 1s/step - loss: 0.0683 - accuracy: 0.9805
Epoch 4/5
30/30 [=====] - 32s 1s/step - loss: 0.0319 - accuracy: 0.9910
Epoch 5/5
30/30 [=====] - 32s 1s/step - loss: 0.0239 - accuracy: 0.9945
<keras.callbacks.History at 0x7f9ba7f56b50>

```



```
model.save
```

```

<bound method Model.save of <keras.engine.sequential.Sequential object at
0x7f9bac8c77d0>>

```

```

test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = utils.pad_sequences(test_sequences,maxlen=max_len)

```

```
accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [=====] - 4s 85ms/step - loss: 0.0737 - accuracy: 0.9821
```



```
l = accr[0]
```

```
a =accr[1]
```

```
print('Test set\n Loss: {:.3f}\n Accuracy: {:.3f}'.format(l,a))
```

```
Test set
```

```
Loss: 0.074
```

```
Accuracy: 0.982
```

Welcome to Colab!

If you're already familiar with Colab, check out this video to learn about interactive tables, the executed code history view, and the command palette.



What is Colab?

Colab, or "Colaboratory", allows you to write and execute Python in your browser, with

- Zero configuration required
- Access to GPUs free of charge
- Easy sharing

Whether you're a **student**, a **data scientist** or an **AI researcher**, Colab can make your work easier.

Watch [Introduction to Colab](#) to learn more, or just get started below!

▼ Getting started

The document you are reading is not a static web page, but an interactive environment called a **Colab notebook** that lets you write and execute code.

For example, here is a **code cell** with a short Python script that computes a value, stores it in a

```
seconds_in_a_day = 24 * 60 * 60
seconds_in_a_day
```

86400

To execute the code in the above cell, select it with a click and then either press the play button to the left of the code, or use the keyboard shortcut "Command/Ctrl+Enter". To edit the code, just click the cell and start editing.

Variables that you define in one cell can later be used in other cells:

```
seconds_in_a_week = 7 * seconds_in_a_day
seconds_in_a_week
```

604800

Colab notebooks allow you to combine **executable code** and **rich text** in a single document, along with **images**, **HTML**, **LaTeX** and more. When you create your own Colab notebooks, they are stored in your Google Drive account. You can easily share your Colab notebooks with co-workers or friends, allowing them to comment on your notebooks or even edit them. To learn more, see [Overview of Colab](#). To create a new Colab notebook you can use the File menu above, or use the following link: [create a new Colab notebook](#).

Colab notebooks are Jupyter notebooks that are hosted by Colab. To learn more about the Jupyter project, see jupyter.org.

▼ Data science

With Colab you can harness the full power of popular Python libraries to analyze and visualize data. The code cell below uses **numpy** to generate some random data, and uses **matplotlib** to visualize it. To edit the code, just click the cell and start editing.

```
import numpy as np
from matplotlib import pyplot as plt

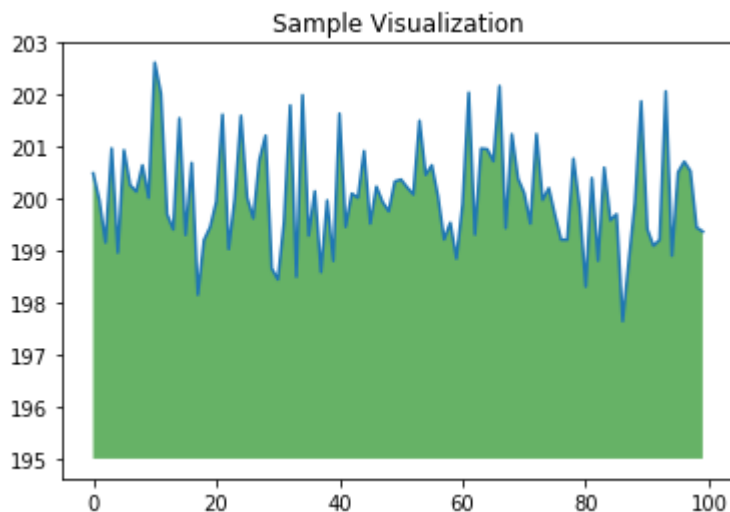
ys = 200 + np.random.randn(100)
x = [x for x in range(len(ys))]
```

```
plt.plot(x, ys, '-')
```

```
plt.fill_between(x, ys, 195, where=(ys > 195), facecolor='g', alpha=0.6)
```

```
plt.title("Sample Visualization")
```

```
plt.show()
```



You can import your own data into Colab notebooks from your Google Drive account, including from spreadsheets, as well as from Github and many other sources. To learn more about importing data, and how Colab can be used for data science, see the links below under [Working with Data](#).

▼ Machine learning

With Colab you can import an image dataset, train an image classifier on it, and evaluate the model, all in just [a few lines of code](#). Colab notebooks execute code on Google's cloud servers, meaning you can leverage the power of Google hardware, including [GPUs and TPUs](#), regardless of the power of your machine. All you need is a browser.


Colab is used extensively in the machine learning community with applications including:

- Getting started with TensorFlow
- Developing and training neural networks
- Experimenting with TPUs
- Disseminating AI research
- Creating tutorials

To see sample Colab notebooks that demonstrate machine learning applications, see the [machine learning examples](#) below.

▼ More Resources

Working with Notebooks in Colab

- [Overview of Colaboratory](#)
- [Guide to Markdown](#)
- [Importing libraries and installing dependencies](#)
- [Saving and loading notebooks in GitHub](#)
- [Interactive forms](#)
- [Interactive widgets](#)
- 

Working with Data

- [Loading data: Drive, Sheets, and Google Cloud Storage](#)
- [Charts: visualizing data](#)
- [Getting started with BigQuery](#)

Machine Learning Crash Course

These are a few of the notebooks from Google's online Machine Learning course. See the [full course website](#) for more.

- [Intro to Pandas DataFrame](#)
- [Linear regression with tf.keras using synthetic data](#)

Using Accelerated Hardware

- [TensorFlow with GPUs](#)
- [TensorFlow with TPUs](#)

Featured examples

- [NeMo Voice Swap](#): Use Nvidia's NeMo conversational AI Toolkit to swap a voice in an audio fragment with a computer generated one.
- [Retraining an Image Classifier](#): Build a Keras model on top of a pre-trained image classifier to distinguish flowers.
- [Text Classification](#): Classify IMDB movie reviews as either *positive* or *negative*.
- [Style Transfer](#): Use deep learning to transfer style between images.
- [Multilingual Universal Sentence Encoder Q&A](#): Use a machine learning model to answer questions from the SQuAD dataset.
- [Video Interpolation](#): Predict what happened in a video between the first and the last frame.

[Colab paid products](#) - [Cancel contracts here](#)

✓ 0s completed at 1:28 PM

