{

"nbformat": 4,

"nbformat\_minor": 0,

"metadata": {

"colab": {

"provenance": [],

"collapsed\_sections": []

},

"kernelspec": {

"name": "python3",

"display\_name": "Python 3"

},

"language\_info": {

"name": "python"

}

},

"cells": [

{

"cell\_type": "markdown",

"source": [

"## \*\*1.Download the dataset\*\*"

],

"metadata": {

"id": "\_gp-BOiCZaUQ"

}

},

{

"cell\_type": "markdown",

"source": [

"## \*\*2.Import required library\*\*"

],

"metadata": {

"id": "bWtCfkZCWxwf"

}

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "mzQjCv-RWWFN"

},

"outputs": [],

"source": [

"import pandas as pd\n",

"import numpy as np\n",

"from sklearn.model\_selection import train\_test\_split\n",

"from sklearn.preprocessing import LabelEncoder\n",

"from keras.models import Model\n",

"from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embedding\n",

"from keras.optimizers import RMSprop\n",

"from keras.preprocessing.text import Tokenizer\n",

"from keras\_preprocessing import sequence\n",

"from keras.utils import to\_categorical\n",

"from keras.models import load\_model"

]

},

{

"cell\_type": "markdown",

"source": [

"## \*\*3.Read Dataset and do preprocessing\*\*"

],

"metadata": {

"id": "-ih2AL3bW7qD"

}

},

{

"cell\_type": "code",

"source": [

"df = pd.read\_csv('spam.csv',delimiter=',',encoding='latin-1')\n",

"df.head()"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 206

},

"id": "Y93XenxDW8DZ",

"outputId": "23759aca-975c-4130-b026-b5e3aa7ab7a0"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

" v1 v2 Unnamed: 2 \\\n",

"0 ham Go until jurong point, crazy.. Available only ... NaN \n",

"1 ham Ok lar... Joking wif u oni... NaN \n",

"2 spam Free entry in 2 a wkly comp to win FA Cup fina... NaN \n",

"3 ham U dun say so early hor... U c already then say... NaN \n",

"4 ham Nah I don't think he goes to usf, he lives aro... NaN \n",

"\n",

" Unnamed: 3 Unnamed: 4 \n",

"0 NaN NaN \n",

"1 NaN NaN \n",

"2 NaN NaN \n",

"3 NaN NaN \n",

"4 NaN NaN "

],

"text/html": [

"\n",

" <div id=\"df-aed1c250-40bb-40d4-89d3-27d1f4a90f43\">\n",

" <div class=\"colab-df-container\">\n",

" <div>\n",

"<style scoped>\n",

" .dataframe tbody tr th:only-of-type {\n",

" vertical-align: middle;\n",

" }\n",

"\n",

" .dataframe tbody tr th {\n",

" vertical-align: top;\n",

" }\n",

"\n",

" .dataframe thead th {\n",

" text-align: right;\n",

" }\n",

"</style>\n",

"<table border=\"1\" class=\"dataframe\">\n",

" <thead>\n",

" <tr style=\"text-align: right;\">\n",

" <th></th>\n",

" <th>v1</th>\n",

" <th>v2</th>\n",

" <th>Unnamed: 2</th>\n",

" <th>Unnamed: 3</th>\n",

" <th>Unnamed: 4</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>ham</td>\n",

" <td>Go until jurong point, crazy.. Available only ...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>ham</td>\n",

" <td>Ok lar... Joking wif u oni...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>spam</td>\n",

" <td>Free entry in 2 a wkly comp to win FA Cup fina...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>3</th>\n",

" <td>ham</td>\n",

" <td>U dun say so early hor... U c already then say...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" <tr>\n",

" <th>4</th>\n",

" <td>ham</td>\n",

" <td>Nah I don't think he goes to usf, he lives aro...</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" <td>NaN</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>\n",

" <button class=\"colab-df-convert\" onclick=\"convertToInteractive('df-aed1c250-40bb-40d4-89d3-27d1f4a90f43')\"\n",

" title=\"Convert this dataframe to an interactive table.\"\n",

" style=\"display:none;\">\n",

" \n",

" <svg xmlns=\"http://www.w3.org/2000/svg\" height=\"24px\"viewBox=\"0 0 24 24\"\n",

" width=\"24px\">\n",

" <path d=\"M0 0h24v24H0V0z\" fill=\"none\"/>\n",

" <path d=\"M18.56 5.44l.94 2.06.94-2.06 2.06-.94-2.06-.94-.94-2.06-.94 2.06-2.06.94zm-11 1L8.5 8.5l.94-2.06 2.06-.94-2.06-.94L8.5 2.5l-.94 2.06-2.06.94zm10 10l.94 2.06.94-2.06 2.06-.94-2.06-.94-.94-2.06-.94 2.06-2.06.94z\"/><path d=\"M17.41 7.96l-1.37-1.37c-.4-.4-.92-.59-1.43-.59-.52 0-1.04.2-1.43.59L10.3 9.45l-7.72 7.72c-.78.78-.78 2.05 0 2.83L4 21.41c.39.39.9.59 1.41.59.51 0 1.02-.2 1.41-.59l7.78-7.78 2.81-2.81c.8-.78.8-2.07 0-2.86zM5.41 20L4 18.59l7.72-7.72 1.47 1.35L5.41 20z\"/>\n",

" </svg>\n",

" </button>\n",

" \n",

" <style>\n",

" .colab-df-container {\n",

" display:flex;\n",

" flex-wrap:wrap;\n",

" gap: 12px;\n",

" }\n",

"\n",

" .colab-df-convert {\n",

" background-color: #E8F0FE;\n",

" border: none;\n",

" border-radius: 50%;\n",

" cursor: pointer;\n",

" display: none;\n",

" fill: #1967D2;\n",

" height: 32px;\n",

" padding: 0 0 0 0;\n",

" width: 32px;\n",

" }\n",

"\n",

" .colab-df-convert:hover {\n",

" background-color: #E2EBFA;\n",

" box-shadow: 0px 1px 2px rgba(60, 64, 67, 0.3), 0px 1px 3px 1px rgba(60, 64, 67, 0.15);\n",

" fill: #174EA6;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert {\n",

" background-color: #3B4455;\n",

" fill: #D2E3FC;\n",

" }\n",

"\n",

" [theme=dark] .colab-df-convert:hover {\n",

" background-color: #434B5C;\n",

" box-shadow: 0px 1px 3px 1px rgba(0, 0, 0, 0.15);\n",

" filter: drop-shadow(0px 1px 2px rgba(0, 0, 0, 0.3));\n",

" fill: #FFFFFF;\n",

" }\n",

" </style>\n",

"\n",

" <script>\n",

" const buttonEl =\n",

" document.querySelector('#df-aed1c250-40bb-40d4-89d3-27d1f4a90f43 button.colab-df-convert');\n",

" buttonEl.style.display =\n",

" google.colab.kernel.accessAllowed ? 'block' : 'none';\n",

"\n",

" async function convertToInteractive(key) {\n",

" const element = document.querySelector('#df-aed1c250-40bb-40d4-89d3-27d1f4a90f43');\n",

" const dataTable =\n",

" await google.colab.kernel.invokeFunction('convertToInteractive',\n",

" [key], {});\n",

" if (!dataTable) return;\n",

"\n",

" const docLinkHtml = 'Like what you see? Visit the ' +\n",

" '<a target=\"\_blank\" href=https://colab.research.google.com/notebooks/data\_table.ipynb>data table notebook</a>'\n",

" + ' to learn more about interactive tables.';\n",

" element.innerHTML = '';\n",

" dataTable['output\_type'] = 'display\_data';\n",

" await google.colab.output.renderOutput(dataTable, element);\n",

" const docLink = document.createElement('div');\n",

" docLink.innerHTML = docLinkHtml;\n",

" element.appendChild(docLink);\n",

" }\n",

" </script>\n",

" </div>\n",

" </div>\n",

" "

]

},

"metadata": {},

"execution\_count": 27

}

]

},

{

"cell\_type": "code",

"source": [

"df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True) #dropping unwanted columns\n",

"df.info()"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "Jp9dyEsyXIi\_",

"outputId": "5b2b9580-a7b8-4ee6-8a79-358c86251be0"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"<class 'pandas.core.frame.DataFrame'>\n",

"RangeIndex: 5572 entries, 0 to 5571\n",

"Data columns (total 2 columns):\n",

" # Column Non-Null Count Dtype \n",

"--- ------ -------------- ----- \n",

" 0 v1 5572 non-null object\n",

" 1 v2 5572 non-null object\n",

"dtypes: object(2)\n",

"memory usage: 87.2+ KB\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"# Count of Spam and Ham values\n",

"df.groupby(['v1']).size()"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "5hHP\_aflXMsq",

"outputId": "bb94fb42-4b9c-4d58-bb3e-a6c09c3430ee"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"v1\n",

"ham 4825\n",

"spam 747\n",

"dtype: int64"

]

},

"metadata": {},

"execution\_count": 29

}

]

},

{

"cell\_type": "code",

"source": [

"# Label Encoding target column\n",

"X = df.v2\n",

"Y = df.v1\n",

"le = LabelEncoder()\n",

"Y = le.fit\_transform(Y)\n",

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

],

"metadata": {

"id": "1bAwbuBMXYld"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"# Test and train split\n",

"X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.15)"

],

"metadata": {

"id": "xLeuDwytXdXg"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"# Tokenisation function\n",

"max\_words = 1000\n",

"max\_len = 150\n",

"tok = Tokenizer(num\_words=max\_words)\n",

"tok.fit\_on\_texts(X\_train)\n",

"sequences = tok.texts\_to\_sequences(X\_train)\n",

"\n",

"sequences\_matrix = sequence.pad\_sequences(sequences,maxlen=max\_len)"

],

"metadata": {

"id": "zTBsIdW0XglM"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## \*\*4.Create Model and 5. Add Layers (LSTM, Dense-(Hidden Layers), Output)\*\*"

],

"metadata": {

"id": "7Uzrg9yCXltD"

}

},

{

"cell\_type": "code",

"source": [

"# Creating LSTM model\n",

"inputs = Input(name='InputLayer',shape=[max\_len])\n",

"layer = Embedding(max\_words,50,input\_length=max\_len)(inputs)\n",

"layer = LSTM(64)(layer)\n",

"layer = Dense(256,name='FullyConnectedLayer1')(layer)\n",

"layer = Activation('relu')(layer)\n",

"layer = Dropout(0.5)(layer)\n",

"layer = Dense(1,name='OutputLayer')(layer)\n",

"layer = Activation('sigmoid')(layer)"

],

"metadata": {

"id": "oQOwFVoTXwsj"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"## \*\*6.Compile the model\*\*"

],

"metadata": {

"id": "KQQV4gQsX3hU"

}

},

{

"cell\_type": "code",

"source": [

"model = Model(inputs=inputs,outputs=layer)\n",

"model.summary()\n",

"model.compile(loss='binary\_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "xxJnjFhvX7Cn",

"outputId": "660a87a0-4389-4384-e3a0-a1046dce263c"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Model: \"model\_2\"\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n",

" Layer (type) Output Shape Param # \n",

"=================================================================\n",

" InputLayer (InputLayer) [(None, 150)] 0 \n",

" \n",

" embedding\_2 (Embedding) (None, 150, 50) 50000 \n",

" \n",

" lstm\_2 (LSTM) (None, 64) 29440 \n",

" \n",

" FullyConnectedLayer1 (Dense (None, 256) 16640 \n",

" ) \n",

" \n",

" activation\_4 (Activation) (None, 256) 0 \n",

" \n",

" dropout\_2 (Dropout) (None, 256) 0 \n",

" \n",

" OutputLayer (Dense) (None, 1) 257 \n",

" \n",

" activation\_5 (Activation) (None, 1) 0 \n",

" \n",

"=================================================================\n",

"Total params: 96,337\n",

"Trainable params: 96,337\n",

"Non-trainable params: 0\n",

"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"## \*\*7.Fit the Model\*\*"

],

"metadata": {

"id": "iKcZ5FNRYKbW"

}

},

{

"cell\_type": "code",

"source": [

"model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=10,\n",

" validation\_split=0.2)"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "x4c4oVp\_YOnO",

"outputId": "628673fa-638e-4718-d58d-3c1351e4aac8"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Epoch 1/10\n",

"30/30 [==============================] - ETA: 0s - loss: 0.3374 - accuracy: 0.8712"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*8.Save the Model\*\*"

],

"metadata": {

"id": "DskMbegoYl3Z"

}

},

{

"cell\_type": "code",

"source": [

"model.save(\"model\_1\")"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "cOQuer-yYoWY",

"outputId": "37034140-29eb-455c-a294-d407b4769166"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stderr",

"text": [

"WARNING:absl:Function `\_wrapped\_model` contains input name(s) InputLayer with unsupported characters which will be renamed to inputlayer in the SavedModel.\n",

"WARNING:absl:Found untraced functions such as lstm\_cell\_1\_layer\_call\_fn, lstm\_cell\_1\_layer\_call\_and\_return\_conditional\_losses while saving (showing 2 of 2). These functions will not be directly callable after loading.\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"# \*\*9.Test the model\*\*"

],

"metadata": {

"id": "Vyt3UFQMY0Ev"

}

},

{

"cell\_type": "code",

"source": [

"test\_sequences = tok.texts\_to\_sequences(X\_test)\n",

"test\_sequences\_matrix = sequence.pad\_sequences(test\_sequences,maxlen=max\_len)"

],

"metadata": {

"id": "WLEzJrA6Y2IN"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"accuracy = model.evaluate(test\_sequences\_matrix,Y\_test)\n",

"print('Accuracy: {:0.3f}'.format(accuracy[1]))"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "2wd-cf0FY9Zx",

"outputId": "e83aeb06-8271-40e8-f3f2-dfd52dcafb23"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"27/27 [==============================] - 1s 36ms/step - loss: 0.1163 - accuracy: 0.9856\n",

"Accuracy: 0.986\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"y\_pred = model.predict(test\_sequences\_matrix)\n",

"print(y\_pred[25:40].round(3))"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "0hcxr-cbZG7l",

"outputId": "f0b1ae71-d6f8-495b-c529-58f90c19d0ea"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"27/27 [==============================] - 1s 20ms/step\n",

"[[0. ]\n",

" [0. ]\n",

" [0. ]\n",

" [0. ]\n",

" [0. ]\n",

" [0.002]\n",

" [0. ]\n",

" [0.024]\n",

" [0. ]\n",

" [0. ]\n",

" [0. ]\n",

" [0. ]\n",

" [0. ]\n",

" [0. ]\n",

" [0. ]]\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"print(Y\_test[25:40])"

],

"metadata": {

"colab": {

"base\_uri": "https://localhost:8080/"

},

"id": "ULiccp6UZNR9",

"outputId": "cf0b0c65-a7c1-46bb-e3ef-ba25e02d1bc0"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"[[0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]\n",

" [0]]\n"

]

}

]

}

]

}