{

"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": [

"#SMS SPAM CLASSIFICATION"

],

"metadata": {

"id": "7dr4Y\_4zmgtw"

}

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "9m70QoiD4te8"

},

"outputs": [],

"source": [

"import pandas as pd\n",

"import numpy as np\n",

"import matplotlib.pyplot as plt\n",

"import seaborn as sns\n",

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

"from sklearn.preprocessing import LabelEncoder"

]

},

{

"cell\_type": "code",

"source": [

"from tensorflow.keras.models import Model\n",

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

"from tensorflow.keras.optimizers import RMSprop\n",

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

"from tensorflow.keras.preprocessing import sequence\n",

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

"from tensorflow.keras.callbacks import EarlyStopping\n",

"%matplotlib inline"

],

"metadata": {

"id": "iqa-aPPYmXrh"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#READ DATASET AND DO PREPROCESSING"

],

"metadata": {

"id": "ORneqxWDmnb3"

}

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"id": "7YBpWoSomZOm"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"df.head()"

],

"metadata": {

"colab": {

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

"height": 206

},

"id": "zxpkPWrNm71P",

"outputId": "eaca1811-788a-403a-a230-7024d353062a"

},

"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-577ee091-d71d-415d-9916-1136b41c8cd7\">\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-577ee091-d71d-415d-9916-1136b41c8cd7')\"\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-577ee091-d71d-415d-9916-1136b41c8cd7 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-577ee091-d71d-415d-9916-1136b41c8cd7');\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": 4

}

]

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"id": "\_XZ6Z1l1m-Y7"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"df.info()"

],

"metadata": {

"colab": {

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

},

"id": "Vvak\_H3QnDNS",

"outputId": "5d05ceee-0bc3-44c6-9ab5-c2beeaa4c28a"

},

"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": [

"sns.countplot(df.v1)\n",

"plt.xlabel('x-axis')\n",

"plt.title('Number of ham and spam messages')"

],

"metadata": {

"colab": {

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

"height": 367

},

"id": "QN20mKt5nHHt",

"outputId": "aa1888ed-b127-4ed1-923c-3154767acbfb"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stderr",

"text": [

"/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.\n",

" FutureWarning\n"

]

},

{

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

"data": {

"text/plain": [

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

]

},

"metadata": {},

"execution\_count": 7

},

{

"output\_type": "display\_data",

"data": {

"text/plain": [

"<Figure size 432x288 with 1 Axes>"

],

"image/png": "\n"

},

"metadata": {

"needs\_background": "light"

}

}

]

},

{

"cell\_type": "markdown",

"source": [

"#CREATE INPUT VECTORS AND PROCESS LABELS"

],

"metadata": {

"id": "g0eRjvCXnuCk"

}

},

{

"cell\_type": "code",

"source": [

"X = df.v2\n",

"Y = df.v1"

],

"metadata": {

"id": "4TxedAC\_nZQU"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"le = LabelEncoder()\n",

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

],

"metadata": {

"id": "uHiIt4YIn\_Bp"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"id": "hc1F-8uXoCI2"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#SPLIT THE TRAINING AND TESTING DATA"

],

"metadata": {

"id": "An7KQLxHoGME"

}

},

{

"cell\_type": "code",

"source": [

"X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.20)"

],

"metadata": {

"id": "OM0Xc87PoFAr"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#PROCESS THE DATA"

],

"metadata": {

"id": "sT63I2sHoTTZ"

}

},

{

"cell\_type": "code",

"source": [

"max\_words = 1000\n",

"max\_len = 150"

],

"metadata": {

"id": "r5LFLEKooR7i"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

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

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

],

"metadata": {

"id": "cs6emJYtoaZ\_"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

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

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

],

"metadata": {

"id": "qlvTMv4fodmK"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#CREATE MODELS AND ADD LAYERS"

],

"metadata": {

"id": "bPhvENzeoiLD"

}

},

{

"cell\_type": "code",

"source": [

"def RNN():\n",

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

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

" layer = LSTM(128)(layer)\n",

" layer = Dense(256,name='FC1')(layer)\n",

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

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

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

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

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

" return model"

],

"metadata": {

"id": "U-SZTrL4og1z"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"model = RNN()"

],

"metadata": {

"id": "5v919ljros4m"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"model.summary()"

],

"metadata": {

"colab": {

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

},

"id": "\_xcI47oRow6y",

"outputId": "639debc2-94d7-4804-9eb6-f3d618d425e2"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

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

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

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

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

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

" \n",

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

" \n",

" lstm (LSTM) (None, 128) 91648 \n",

" \n",

" FC1 (Dense) (None, 256) 33024 \n",

" \n",

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

" \n",

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

" \n",

" out\_layer (Dense) (None, 1) 257 \n",

" \n",

" activation\_1 (Activation) (None, 1) 0 \n",

" \n",

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

"Total params: 174,929\n",

"Trainable params: 174,929\n",

"Non-trainable params: 0\n",

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

]

}

]

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"id": "\_rWF84QJo06N"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#FIT THE MODEL"

],

"metadata": {

"id": "\_Buf6mBko9Sb"

}

},

{

"cell\_type": "code",

"source": [

"model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=100,\n",

" validation\_split=0.2,callbacks=[EarlyStopping(monitor='val\_loss',min\_delta=0.0001)])"

],

"metadata": {

"colab": {

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

},

"id": "10J3H1o0o5v\_",

"outputId": "0a5b2ccb-959b-4deb-c39e-d013f62acb33"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Epoch 1/100\n",

"28/28 [==============================] - 13s 460ms/step - loss: 0.0961 - accuracy: 0.9778 - mse: 0.0358 - mae: 0.1438 - val\_loss: 0.1271 - val\_accuracy: 0.9832 - val\_mse: 0.0568 - val\_mae: 0.2060\n",

"Epoch 2/100\n",

"28/28 [==============================] - 14s 507ms/step - loss: 0.0728 - accuracy: 0.9885 - mse: 0.0607 - mae: 0.2129 - val\_loss: 0.1175 - val\_accuracy: 0.9821 - val\_mse: 0.0766 - val\_mae: 0.2416\n"

]

},

{

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

"data": {

"text/plain": [

"<keras.callbacks.History at 0x7f2c8029f350>"

]

},

"metadata": {},

"execution\_count": 20

}

]

},

{

"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": "aZrhpoCvpFSh"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

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

],

"metadata": {

"colab": {

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

},

"id": "UcKyVUwcpjrH",

"outputId": "509e119d-dcdc-40df-a551-294372b0f7fe"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"35/35 [==============================] - 3s 92ms/step - loss: 0.1390 - accuracy: 0.9821 - mse: 0.0779 - mae: 0.2393\n"

]

}

]

},

{

"cell\_type": "code",

"source": [

"print('Test set\\n Loss: {:0.3f}\\n Accuracy: {:0.3f}'.format(accr[0],accr[1]))"

],

"metadata": {

"colab": {

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

},

"id": "A1P0obznpoWE",

"outputId": "3e35375d-8ea8-449e-b836-8cbbb5eff307"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Test set\n",

" Loss: 0.139\n",

" Accuracy: 0.982\n"

]

}

]

},

{

"cell\_type": "markdown",

"source": [

"#SAVE THE MODEL"

],

"metadata": {

"id": "LpuLRZbJpuaW"

}

},

{

"cell\_type": "code",

"source": [

"model.save(r\"C:\\Users\\vikram\\OneDrive\\Desktop\\model\_lSTM.h5\")"

],

"metadata": {

"id": "d1WBN2czpsue"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "markdown",

"source": [

"#TEST THE MODEL\n"

],

"metadata": {

"id": "rrHj3HxCqT9b"

}

},

{

"cell\_type": "code",

"source": [

"from tensorflow.keras.models import load\_model\n",

"m2 = load\_model(r\"C:\\Users\\vikram\\OneDrive\\Desktop\\model\_lSTM.h5\")"

],

"metadata": {

"id": "7aSPr\_6RqDjX"

},

"execution\_count": null,

"outputs": []

},

{

"cell\_type": "code",

"source": [

"m2.evaluate(test\_sequences\_matrix,Y\_test)"

],

"metadata": {

"colab": {

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

},

"id": "UfXAv0N2qgl2",

"outputId": "66d0c1c3-728b-40a4-85e2-6a5c10ed4e09"

},

"execution\_count": null,

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"35/35 [==============================] - 4s 68ms/step - loss: 0.1390 - accuracy: 0.9821 - mse: 0.0779 - mae: 0.2393\n"

]

},

{

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

"data": {

"text/plain": [

"[0.13899557292461395,\n",

" 0.9820627570152283,\n",

" 0.07788368314504623,\n",

" 0.23931345343589783]"

]

},

"metadata": {},

"execution\_count": 26

}

]

},

{

"cell\_type": "code",

"source": [],

"metadata": {

"id": "f5vO2h97qrC5"

},

"execution\_count": null,

"outputs": []

}

]

}