# ASSIGNMENT-4

**NAME: HARNISRIE L**

**ROLL NUMBER: 714019106030**

**COLLEGE NAME: SRI SHAKTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY ,COIMBATORE.**

# SMS SPAM CLASSIFICATION

```
{
  "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": [
       "# **2. IMPORT LIBRARIES**"
     ],
     "metadata": {
       "id": "h206JVIxjJqt"
     }
   },
```

```json
{
  "cell_type": "code",
```

```
    "execution_count": null,

    "metadata": {

      "id": "jTLGPy_fgQJC"

    },

    "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\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.callbacks import EarlyStopping\n",

      "from keras.utils import pad_sequences\n",

      "%matplotlib inline"

    ]

  },

  {

    "cell_type": "markdown",

    "source": [

      "# **3.1. READ DATASET**"

    ],

    "metadata": {

      "id": "w2-i5_IGjSUh"

    }
```

```
},
{
 "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": "b2S-rygKh48X",
  "outputId": "bb6b8163-f10b-4479-e328-681429a9c109"
 },
 "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-f2140a1b-e52d-4a73-807f-7b169f50a515\">\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-f2140a1b-e52d-4a73-807f-7b169f50a515')\"\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-f2140a1b-e52d-4a73-807f-7b169f50a515 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-f2140a1b-e52d-4a73-807f-
7b169f50a515');\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": 2
  }
 ]
},
{
  "cell_type": "markdown",
  "source": [
    "# **3.2. PREPROCESSING**"
  ],
  "metadata": {
    "id": "RzavY7S-jiO2"
  }
},
{
  "cell_type": "code",
  "source": [
    "df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)\n",
    "df.info()"
  ],
  "metadata": {
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
```

```json
      "id": "UXZj9dv9h-Lq",
      "outputId": "003ad4bf-a18e-4a95-a237-635d94e25ea9"
    },
    "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('Label')\n",
      "plt.title('Number of ham and spam messages')\n",
      "X = df.v2\n",
      "Y = df.v1\n",
      "le = LabelEncoder()\n",
```

```
      "Y = le.fit_transform(Y)\n",

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

     ],

     "metadata": {

      "colab": {

       "base_uri": "https://localhost:8080/",

       "height": 351

      },

      "id": "JqQHhqtuiCbC",

      "outputId": "de038311-bc99-46d1-af4a-fe897347073c"

     },

     "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": "display_data",

       "data": {

        "text/plain": [

         "<Figure size 432x288 with 1 Axes>"

        ],

        "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAYsAAAEWCAYAAACXGLsWAAAABHNCSVQICAgIfAhkiAAAAlwSFlz
AAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA
```

6Ly9tYXRwbG90bGliLm9yZy+WH4yJAAAZjUlEQVR4nO3deZhddZ3n8feHBEQFUhESGjDKLbirhGwtbtp
HAFxgcd2wXEJimI72toz7do9I4j4qK0j7nbTioCOIu5p2xZRcRtFSFxAQCXDYYsKWSAKCoiPwnT/Or+RQVO
VUILeqknq/nuc+dc7vLPd7zr11P/esN1WFFEkbs81MFyBJmv0MC0nSIAkDTIsJEmDDAtJ0iDDAtJ0iDD
QlOS5KQkx83QcyfJR5NNsSHL2BMPSPLdmahtc0qyf5I1M12HNBHDAyRuguV5NIka5Pcvdf24iTfnGGyRuXx
wBOBxVW1z0wXI81FhsWWbR7wqpkuYlMlmbeJk9wXuKaqkuQNyy2bJk9wXuLSqfjOKeiQNMyy2bO8AXp1kp/EDkixJUknm99
q+meTFrfulJP8nyfFFJrk1ycZI/a+2r21bLsnGz3SSXJGUmuT/KtJPftzfuBbqzeJOJSfCjJl5P8BvirCerd
PcnyNv2qJC9p7UcCHwYYem+SGJG+abGGUkeWfbVXJJkif12l+Y5MJW98VJXjobtn+yiSHJTkkyyyYQ
9aAPf+wked8cpIfJfJ1W2/HTPAaLEvyyyyS/SvKPveF3betmQ5ILgMds5HnSXxqu17nbOOS/KQ3vr95428/9JMn+v
WFHtNfn+vYaPre1378t03VtHX5Xqqst21yQnt/V7YYn1bnOS/KQfjee8/9JMn+v
WFHtNfn+vYaPre1378t03VtHX5Xqqst21yQnt/V7YYn1bnOS/KQfjee8/9JMn+v
f06ycokf94bdkySTyf5eJv2vCQPSfz1eJv2vCQPSfzfjaDi3vMaSP3u95428Nu9pt
WFHtNfn+vYaPre1378t03VtHX5Xqqst21yQnt/V7YYuN1/SG757ks0nWted7ZW/YYPkkWtPleneRdk62DOa
OqfGyBD+BS4D8DnwOOa20vBr7ZupcABzvfNN4MWt+wjgJUUELge2KGNf1r/gC8pC3Ly4
ArgLThTwbuBWT4S+C3wKPasP3bengjsG2bxzrgE8wOwlOBG4E9J3nu/YVOG/KfAmkme5yBgJbBTW44o41
HpYXNQb9tA2V737YYD/BFwMHNSGAsbX09sfUvpHsP/Rr40zbubQAhbBX7tj6bfNa2V737YD/BFwMHNSGfx94fuveAdhvpv/nZ/ox4wX4wX4uIMv3K1h8RC6D+KFb
HpYXNQb9tA2/q69tmuAR7Tuk4BTe8N2AG6m+5B7NvNvdcfX9S+9D/9D4iTglI0syx5tXjXjv22t4KnNSrdSgSDgSvVvX
679aW5T6TjjP8F4FWe3+6MJjjX+nds0+7bG38lLQCm8Lq8Gzh+3+3+3GuwuDf8bODw1n0xcHBv2FYHA
L8A9gO2GTds0tdmknltAB7euo8BzugNeypwwwTrY6eNvA+f2+v/LPChXv/fAl9o3o3a8DPjZu+tOBZXRhcS3w18Bdx41zCnBBCfz1uZZ33+v/LPwdf
w18Bdx41zCnBBCfz1uZP33l+2PH/6t/8XcGhb7Ar8cN+bgI+27g9DbwJ22Zz/t1vyw91QW7iq+inwJeD/bgd41d
2Dyq3vdN7b5jW/bode/uve8NwDrgd3pjins23YtXKvkWuC5wH0mmnCCuwXRnmX6ydX9/mN5sPPrq+r6XTtldN9/b
ZZ07ACR5UpKz2i6la+m+m+2e7Sm/aaaH/M24ed+mNrqtmrF57c5t18t18tlky1
cVX0DeD/dlt/aJCckUdvlMleG5k8uu2Gua4t/z3H1Th+WX81wfqYYcPknm9tC+g3g3dl46
/Aa5M8u9JHtimey3d1tTZSc5P8PoqKxmQ8s8/s12+++L7D7uFr+gW4LCeBI4AHAz5Kck+Qk8/j12+++L7D7uFr
cTbe7oP/hOnYw+G69tv7H9x2xx2xx1hHkh3odp9cQfdP+K2q2q2PwN74C2DnJjr22PwEuv5P
1kuQudN9230m31bQT8GW6D5/N4RPAcrpv8lbRT8yTqqr3VtWjjb5p3gb3pPshe7x1
/FngWsKAt/3FbU0Z/Vrl7Vb0NoKpOr6on0OC/hnd7on0u2C+hnd7uzJjgb3pPshe0xs84WvT9uG
/GtNf16vpdq/1a9xqg5/pz3lRVT0HuDfZFuB4oFV038u1+
GtNf16vpdq/1a9xqg5/pz3lRVT0HuDfZFuBwiO038bus
gnzfg7dPv4rgM/THe/42p2sd6y2V9J9cGyg+zBffmf2/NfgWOTXE93sPS0TXS7TXS7ni4Bvgp8bCPj3oPu
m/aGNs01dGfFjDjZnwtaE7JvAVuuMdl9GFzMZ2MC69rGGt1vIbuc2gb4L/Tvf7r6U05EGNsyfQzwg
yQ30L12r6qqixletmOBNXTr92vAZ+iCiLab7Sl0zPXXTHmW5kDhs7W0TSFi
rJSXQHbv/HTNcymyV5Gd2H/l/OdC1bIcsyW5Gd2H/l/OdC1brcsJ2Vkuya5+puaUs0fHkkyp3Sn1n5+puaUs0fHkkyp3Cvcfdc
YdT6a7l0R3gbihJ0iB3Q0mSBo10N1SSS+luQ3AzcdFNVLU2yM91pLd8mSy4O4w0w0TSJi
rJSXQHbv/HTNcymyV5Gd2H/l/OdC1brcsJ2VkuyY5+puaUs0fHkkWStkjb0Z3CvSfdc
YdT6a7l0R3gbihJ0iB3Q0mSBo10N1SSS+luQ3AzcdFNVLU2yM91pnkvorv58VlVtSBK6sw4Oobtw6Yiq+m
GbzzJg7ODdcVV18saed5dddqklS5Zs9uWRpK3pZ3ZypUrf1VVCycaNlt7/6l0S3cxD7A7A7A7A7A7A7
PAvzqj32BD9FdFbwz3SmBS+nOr16ZZZX1VVCycaNh2+++L7D7uFr+gW4LCeBI4AHAz5Kck+Qk8/j12+++L7D7u
U6pzFrBTkt3obnB2RlWtbwFxt050JkKaTLqsCjgq+3WwUe1tl2r6srWfRW33otlEbe9oGZNa5us/TaSHN
VuKbxi3bp1m3MZJGnOG6jn/FdJGnn, OG/VuqMdX1eVJ7g2ckeRn/YFVVVUk2y+y+lYVXUC3Z0pWbp0qdti6q6v
P1dS3cxD7A1W33Eu3v2jb65dz6Rl+LW9tk7k7iLabgx1I9+Muy+nuX0X0/+8XWvRx4QTr
7Ade13VWnAwe2m4EtaPM5fVR+oXY7iLabgx1I9+Muy+nuX0X0/+8XWvRx4QTr
7Ade13VWnAwe2m4EtaPM5fVR1S5Jub5S7oXYFPt+dEct84BNV9ZUk5wCnpufi8zMvbizjDwFW
0Z06+0KAqlqf5M3AOW28Y6tq/fx6NecMtMlaBZa+Y4XzHQQ0ooxwy0KSNGgeQW3JGgmQYSFJuRdZfx6
8IMmqJJ9Ksl1rv0omGeQW3JGgmQYSFJuRdZfx6
JA0yLCRJgwwLSdlgwwLSdIgw0KSNMiwkSYl+VGSL7X+VGSL7X+PZP
8IMmqJJ9Ksl1rv0omGeQW3JGgmQYSFJuRdZfx6
NecMtMlaBZa+Y4XzHQQ0oxwy0KSNMiwkCQNMiwkSYMMC0nSIMNCkjTIsJAkDTIsJEkDRh4WSeYl+VGSL7X+PZP
8IMmqJJ9Ksl1rv0vrX9WGL+nN4w2t/t8OHF9V9w6wc2Aee29hKk25zKk25qOLYtXARf2+t8PHF9V9w6wc2AEe29iOBDa39+DYe

SfYGDgceDBwMfDDJvGmoW5LUjDQskiwGngx8uPUHOAD4TBvlZOCw1n1o66cNf0Ib/1Dg1Kr6fVVdAq
wC9hll3ZKk2xr1lsW7gdcCt7T+ewHXVtVNrX8NsKh1LwJWA7Th17Xx/9g+wTR/lOSoJCuSrFi3bt3mXg5J
mtNGFhZJngKsraqVo3qOvqo6oaqWVtXShQsXTsdTStKcMX+E834c8LQkhwDbA/cA3gPslGR+23pYDFze
xr8c2ANYk2Q+cE/gml77mP40kqRpMLIti6p6Q1UtrqoldAeov1FVzwXOBJ7RRIsGfLF1L2/9tOHfqKpq7Ye
3s6X2BPYCzh5V3ZKk2xvllsVkXgecmuQ44EfAR1r7R4CPJVkFrKcLGKrq/CSnARcANwEvr6qbp79sSZq7pi
UsquqbwDdb98VMcDZTVf0OeOYk078FeMvoKpQkbYxXcEuSBhkWkqRBhoUkaZBhIUkaZFhIkgYZFpKk
QYaFJGmQYSFJGmRYSJJIGGRaSpEGGhSRpkGEhSRpkWEiSBhkWkqRBhoUkaZBhIUkaZFhIkgYZFpKkQYa
FJGmQYSFJGmRYSJJIGGRaSpEGGhSRpkGEhSRpkWEiSBhkWkqRBhoUkaZBhIUkaZFhIkgYZFpKkQYaFJG
mQYSFJGmRYSJJIGGRaSpEEjC4sk2yc5O8lPkpyf5E2tfc8kP0iyKsmnkmzX2u/S+le14Ut683pDa/95koNGV
bMkaWKj3LL4PXBAVT0ceARwcJL9gCL9gLcDx1fV7cDtwJzrhdV/YENwJFt/COBDa39+DYeSfYGDgceDBwMfDDJvBHWLUka
Z2RhUZ0bWu+27VHAAcBnWvvJwGt27VHAAcBnWvvJwGt27VHAAcBnWvvJwGt27VHAAcBnWvvJwGGt+9DWTxv+hCRp7adW1e+r6hJgfbDPqOqWJN3eSl9ZJJmX5MfA
WuAM4P8C11bVTW2UNCci1r0IWA3Qhl8H3Qhl8H3KvfPsE0/ec6KsmKJCvWrVs3iWRpDlrpGFRVTdX1SOAxwL7Aw4DsAxXR
bAw8c4XOdUUFVLq2rpwoWoULR/U0kjQnTcvZUFV1LXAm8DhgzyQLWtdewOpXu3VLOAm8FhpvyTz26DFowWt+3Jgd4A4A2/AbgeJeXlU3j7j7BuSdI4IwuLJAuT7Nmed4A2ASYQJJmzST73b
zl7YBTquqLyW5ADgtyT8B59zcl7YBT+qqqLyW5ALgtyT/AnwsySwDg1yDPE/AnwsySpgpd0ZUFFTV+UlOAy4IIguLJAuT9J07BuSdI4IDDQpI0yLCQJA2aUlgk+fpU2iRJc9OUwiLJA2aUlg
uAtEH3YIKrqCVJW6ehs6DuvwwmbgD0tfA+0dYlyRpFtloWFRVW9erpo21LGFT+4dYlYkwFFTVe4M2NSwmMuJ4zDJPVYW9
L8mfAUv601TVKSOqS5I0i0i0wplJ8i0wpLJJ8DLgf8AkM+8DLgf8AngY8gzJ7mfS4rpAgWLSJj8pUw+KYURhSZrcdpno21LdGXYgkafaa6tlQ19gG4Dgf1GVZQka
vqW5Z7ZJAgkajaQ1lyIRJs8pUw+KYURhSZrcdpno21LdGXYgkafaa6tlQ19gGgdgf1GVZQka
seoCpMkzR5T3bLYcaw7SYBDgf1GVZQkaXbZ5LvOVucLwEEjqEeSNAtNdTfU03u929Bdd/G7kVQkSZp1
no21FN73TcBl9LtipIkzQFTPWbxwlEXIkmavab0pHWShwC7Ap9uX/0eLk3a3w2+ydr+GySsaTpI0O0z1
mCVvb+L9laIZNtPl6yiVcJ53WK72SaVvHvs1L/nB6JvHHhf7CPWE5PbgVXvI8DTSrkAdnEJIBKk
wBHjKgmSdIsMNXdUE9v929Bdd/G7kVQkSZp1RBoWkaRBhoUkaZBhIUkaZFhIkgYZFpKkQYaFJGmQYaFJG
mQYSFJGmRYSJJIGGRaSpEGGhSRpkGEhSRpkWEiSBhkWkqRBW0xYJdk4yc+TrEzylU6mZBhkWkqRBW0xYJdk4yc+TrEzylU6m

z4APAnYG3hOkr1tipJmju2iLAA
z4APAnYG3hOkr1ntipJmju2iLAA9gFWVdXVFVfX/gFOBQ2e4etFTtAhY3Y3etFuAXOB4oiUcBRrfe
GJD+fptrmgl2AX810EbNB3rlspkvQbfneHHN0Nsdc7jvZgC0lLAZV1QnACTNdx9YoyqqqWjrTdUjj+d6cPlv
rXxvTpNU1UzXIEma5baUYhaU3Ema5baUYhaQ3c1iSJUl+OtN1SJLmgJkuQpI0+/0/GuS85N8Ncldk
lJfpLks01SSRpFvtpkBPW7GZaU3VCSpBVm7ko0iSpFppSzCJJmrsMC0nSIMNCkjTIsJAkDTIsJEmD/n/Bh1jL

jyoyRfS7Jraz8myclJvpPksiRP703/lSTbzuxSbrkMC+0FfKCqHgxcC/w18LmqekxVPRy4EDiyN/4C4LHAf6O
71uV44MHAQ5M8Ylor19bmYOCKqnp4VT0E+Eprv66qHgq8H3h3a/susF9VPZLuXnGv7c3nfsABwNOAj
wNntulvBJ48+sXYOhkWuqSqfty6VwJLgIe0b2bnAc+lC4Mx/1bd+dbnAVdX1XlVdQtwfptWuqPOA56Y5
O1J/ryqrmvtn+z9fWzrXgyc3t6jr+G279H/qKo/tPnN49bQOQ/fo3eYYaHf97pvprtQ8yTgFe3b2JuA7ScY/
5Zx097CFnKRp2anqvoF8Ci6D/XjkrxxbFB/tPb3fcD723v0pUzwHm1fYv5Qt15M5nv0TjAsNJEdgSvb/t3nz
nQxmhuS7A78tqo+DryDLjgAnt37+/3WfU9uvT+ctwKeBqasJvI/gR8A69rfHWe2HM0RDwXekeQW4A/A
y4DPAAuSnEu3xfCcNu4xwKeTbAC+Aew5/eXOLd7uQ9KsleRSYGlV+ZsVM8zdUJKkQW5ZSJIGuWUhSRp
kWEiSBhkWkqRBhoV0JyS5YRPGPSbJq0c1f2mUDAtJ0iDDQtrMJrsjavPwJN9PclGSl/SmeU270++5Sd40A
2VLG2VYSJvfxu6I+jC6O6I+Fnhjkt2THEh39999gEcAj07yF9Ncs7RR3u5D2vwWA59KshuwHXBJb9gXq+p
G4MYkZ9IFxOOBA4EftXF2oAuPb09fydLGGRbS5vc+4F1VtTzJ/nT3MRoz/irYAgK8tar+ZXrKkzadu6GkzW
9jd0Q9NMn2Se4F7A+cA5wOvCjJDgBJFiW593QVK02FWxbSnXO3JGt6/e9i43dEPRc4E9gFeHNVXQFcke
RBwPeTANwAPA9YO/rypanx3lCSpEHuhpIkDTIsJEmDDAtJ0iDDQpI0yLCQJA0yLCRJgwwwLSdKg/w+zxO5
JjPQAtAAAAABJRU5ErkJggg==\n"
        },
        "metadata": {
          "needs_background": "light"
        }
      }
    ]
  },
  {
    "cell_type": "code",
    "source": [
      "X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)"
    ],
    "metadata": {
      "id": "KBLBUSHgiGR6"
    },
    "execution_count": null,
    "outputs": []
  },
  {
    "cell_type": "code",
    "source": [
      "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",

    "sequences_matrix = pad_sequences(sequences,maxlen=max_len)"

   ],

   "metadata": {

    "id": "GD0PDa_6iJ5Q"

   },

   "execution_count": null,

   "outputs": []

  },

  {

   "cell_type": "markdown",

   "source": [

    "# **4. CREATE MODEL**"

   ],

   "metadata": {

    "id": "voU0xeukjy2s"

   }

  },

  {

   "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(64)(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('sigmoid')(layer)\n",
   "    model = Model(inputs=inputs,outputs=layer)\n",
   "   return model"
  ],
  "metadata": {
   "id": "19_2oCf7iMnG"
  },
  "execution_count": null,
  "outputs": []
 },
 {
  "cell_type": "markdown",
  "source": [
   "# **5. LSTM LAYERS**"
  ],
  "metadata": {
   "id": "H16MElYIj-6U"
  }
 },
 {
  "cell_type": "code",
  "source": [
   "model = RNN()\n",
   "model.summary()"
  ],
  "metadata": {
   "colab": {
    "base_uri": "https://localhost:8080/"
   },
   "id": "IoSbYBeSiO71",
```

    "outputId": "415360da-252f-4409-e92f-e69b94218c38"
  },
  "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, 64)                29440   \n",
      "                                                              \n",
      " FC1 (Dense)                 (None, 256)               16640   \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: 96,337\n",
      "Trainable params: 96,337\n",

    "Non-trainable params: 0\n",
     "_____\n"
   ]
  }
 ]
},
{
 "cell_type": "markdown",
 "source": [
  "# **6. COMPILING THE MODEL**"
 ],
 "metadata": {
  "id": "AoLKzOk7kKUI"
 }
},
{
 "cell_type": "code",
 "source": [
  "model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])"
 ],
 "metadata": {
  "id": "M6YvLYRPiVKC"
 },
 "execution_count": null,
 "outputs": []
},
{
 "cell_type": "markdown",
 "source": [
  "# **7. FIT THE MODEL**"
 ],

```
    "metadata": {

      "id": "oJ9_Px0ykPR3"

     }

    },

    {

     "cell_type": "code",

     "source": [

"model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,validation_split=0.2,callbacks=[Earl
yStopping(monitor='val_loss',min_delta=0.0001)])"

    ],

    "metadata": {

      "id": "fRyicKkgiYRk",

      "colab": {

       "base_uri": "https://localhost:8080/"

      },

      "outputId": "fa35d0e8-8207-4afc-cd6d-ee9c78221a5a"

    },

    "execution_count": 11,

    "outputs": [

     {

      "output_type": "stream",

      "name": "stdout",

      "text": [

       "Epoch 1/10\n",

       "30/30 [==============================] - 7s 229ms/step - loss: 0.0452 - accuracy:
0.9873 - val_loss: 0.0302 - val_accuracy: 0.9895\n",

       "Epoch 2/10\n",

       "30/30 [==============================] - 7s 230ms/step - loss: 0.0344 - accuracy:
0.9902 - val_loss: 0.0328 - val_accuracy: 0.9916\n"

      ]

     },
```

```json
    {
     "output_type": "execute_result",
     "data": {
      "text/plain": [
       "<keras.callbacks.History at 0x7fa5b3a662d0>"
      ]
     },
     "metadata": {},
     "execution_count": 11
    }
   ]
  },
  {
   "cell_type": "markdown",
   "source": [
    "# **8. SAVE THE MODEL**"
   ],
   "metadata": {
    "id": "PF-_P9vUkVzf"
   }
  },
  {
   "cell_type": "code",
   "source": [
    "model.save('Spam.h5')"
   ],
   "metadata": {
    "id": "E_ThW_8Rih3M"
   },
   "execution_count": 12,
   "outputs": []
```

    },
    {
      "cell_type": "markdown",
      "source": [
        "# **9. TEST THE MODEL**"
      ],
      "metadata": {
        "id": "5RN24hYukYXD"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "test_sequences = tok.texts_to_sequences(X_test)\n",
        "test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)\n",
        "test_sequences_matrix"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "zWAJtip4i3Ip",
        "outputId": "19759c31-6a03-4d24-fa11-30fc97e7e7f6"
      },
      "execution_count": 13,
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "array([[  0,   0,   0, ...,  45,  44, 197],\n",

```
"        [  0,  0,   0, ...,  455,  56, 106],\n",
"        [  0,  0,   0, ...,    2, 171,  41],\n",
"        ...,\n",
"        [  0,   0,   0, ...,  59, 170, 718],\n",
"        [  0,   0,   0, ...,  245,  11, 269],\n",
"        [  0,   0,   0, ...,  153, 267, 224]], dtype=int32)"
    ]
   },
   "metadata": {},
   "execution_count": 13
  }
 ]
},
{
 "cell_type": "code",
 "source": [
  "accr = model.evaluate(test_sequences_matrix,Y_test)\n",
  "print('Accuracy:',accr[1])\n",
  "print('Loss:',accr[0])"
 ],
 "metadata": {
  "colab": {
   "base_uri": "https://localhost:8080/"
  },
  "id": "rXcmXfBKi8J7",
  "outputId": "1457c9a5-08b5-45d8-a0f2-29087cd9da0c"
 },
 "execution_count": 14,
 "outputs": [
  {
   "output_type": "stream",
```

      "name": "stdout",

      "text": [

        "27/27 [==============================] - 1s 21ms/step - loss: 0.0664 - accuracy: 0.9821\n",

        "Accuracy: 0.9820573925971985\n",

        "Loss: 0.06643393635749817\n"

      ]

    }

   ]

  }

 ]

}