# ASSIGNMENT-4

**NAME: HEMANTH M**

**ROLL NUMBER: 714019106033**

**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"
   }
  },
{
  "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/"
    },
    "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":
"iVBORw0KGgoAAAANSUhEUgAAAYsAAAEWCAYAAACXGLsWAAAABHNCSVQICAgIfAhkiAAAAAlwSFlz
AAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yLjIsIGh0dHA
```

6Ly9tYXRwbG90bGliLm9yZy+WH4yJAAAZjUlEQVR4nO3deZhddZ3n8feHBEQFJUhESGjDKLbirhGwtbtp
HAFxgcd2wXEJimI72toz7do9I4j4qK0j7nbTioCOIu5p2xZRcRtFSFxAQCXDYsKWSAKCoiPwnT/Or+RQVO
VUILeqknq/nuc+dc7vLPd7zr11P/esN1WFJEkbs81MFyBJmv0MC0nSIMNCkjTIsJAkDTIsJEmDDAtJ0iDD
QlOS5KQkx83QcyfJR5NsSHL2BMOPSPLdmahtc0qyf5I1M12HNBHDYguV5V5Ika5Pcvdf24iTfnMGyRuXx
wBOBxVW1z0wXI81FhsWWWbR7wqpkuYlMlmbeJk9wXuLSqfjOKeiQNMyy2bO8AXp1kp/EDkixJUknknm99
q+meTFrfuIJP8nyfFJrk1ycZI/a+2r21bLsnGz3SXJJGUmuT/KtJPftzfubd/bdj6JD9P8qzesiZPSfCjJl5P8PvirCerd
PcnyNv2ZqJC9p7UcCHwYem+SGJGUmuT/KtJPftzfuBbdj6JD9P8qzesJOSfCjJl5P8BvirCerd
PcnyNv2qJC9p7UcCHwYem+SGJG+abGUkeWfbVXVJkif12l+Y5MJW98VJXtobtn+SNUle25b5yiSHJTkkyS
9aPf+wked8cpIf/HTPAaLEvyyyS/SvKPveF3betmQ5ILgMds5HnSXqu17bnOS/KQ3vr95428Nu9pt
f06ycokf94bdkySTyf5eJv2vCQPaSvfKaJOcm+U2SjyTZNcl/tPl9LcmC3vx3kSQ3vvj7Jflee8/9JMn+v
WFHtNfn+vYaPre1b5378t03VtHX5qist21yQnt/V7YXuN1/SG757ks0sn0nWted7dk62DOa
OqfGyBD+BS4D8DnwOOa20vBr7ZupcABczFsoxwG/BD4A3AU4ELge2KGNf1Lr/
4s2/D3Ad9uwuwOr27zmA48EfgXs3Zv2OuBxdxvF9Qtp9geb4NfBDYHngEsA44oFfrdzeyLo4A/gC8pC3Ly4
ArgLThTwbuBwT4S+C3wKPasP3bengjsG2bxzrgE8COwIBG4E9J3nu/YGHtuV6GHA1cNi41+BfgbsCDw
d+DzyoDX8b8B1gZ2AP4KfAmkme5yBgJbBTW44HAbsNvAe7XX5u+Bq8ZeA+AY4Hdt/vOBU4BL
gH/srY9LBt6HZwG7AouAtcAP23tge+AbwNFt3EXANcAhbX09sfUUvpHsP/Rr40zbubsCDW/cnWz3btHk+f
orL9jbgW8ACYDFw7tj6bfNa2V737YD/BFwMHNSGfx94fuveAdv+fuveAdhvpv/nZ/ox4wX4uIMv3K1h8RC6D+KFb
HpYXNQb9tA2/q69aa8BHtG6TwRO6Q3bAbmIbeHYkXlnUtR0Kt8i5T3wfQBOHxKl3UI8Dd+ryfX4Tg2Vl0wSGAe
ZeBuUOAgCjl2C7hrn3eBy8GGxUJ99bsb3WBfpxk+7jjM4iuyRQ4kGkXt6/UE7x0oESLm9UKl/pXSH6Bi0z4y8IOUVzvK
R0R7L2uF9KSkRqlp1Q+Ll45B91XKi6Ovp5bmvJ2G+H3C2TmKhrOMh9xMY+FALDAK0rUmeY/lrx5BR4rrNl5WDfHj3w9atdrA2UHCv0g8Ezpi34CX+5nYvNwNjTiIhBUvKR3XhaZjlrsKK2LZFR9LmuQH4P7dMSXrE+pNaqWNzpHZd6tGqdHLCJ7bDAH0hDt0k3+GF/3VmdDRqdH9GQhkPAf0DQzCg/ko4yvfLjbRIc+X3nj7I7U8S8shnKVDXe8ak2i+n8UyKPk1VXBKg+kxd3Wvre2bv+CJyDSsC7aMg4fQXzB6IbIzxqGYKaeBgt7dLBaNTxxUltAV/I5YWSuOxC/SVHU7Ddl2BPmhfaUqZ4hqF20p28N/P3Gid8yNg27BaO5EdbX+hKN2V/BJLsmxN6+uXzoOzBz8qkDfCDRbg2d39DFvTsSDyHGJO0lxHRCvgdpu8X2/KVU1NDyrHVmiFRjfJKXKBDkeD8FaHq7PU8/cUPLwDnFHBk1W0tDqKAhOGc9GCCgFIIfY2ovm0bWaISh2/etjM8rUtaqMxJ8nlUbi5lvLfUzTQrwLpZy+AaNSBGC9/nSPmHJWI+4cHFlJBsq0tNALNAGMx8ckoQuyYoWTg+hbu0jxXnRIRDI5OFDgCB+TP/DeQRNgThMHw0UGAwHy3ImPKCPRNqaXYGHEHXT6sUJP8aZ+xSyV6gJ8tmPLq5Olvn7G9jtoSAy8w8wmsAlBg6wQEKGTHW/ghcuRpakgQSJYxnkQlPVpWkVSIYGRG9bGnUfA64zlKXjQs14s8AQ0+EgBVjBDyUd22hXe7+VZ0qEp4ZkGuzCPUQzRkhrVvj3p6lxtzIb6WmeDiu3YB4Ksq8hE03j2ZBJE2ahrH+jqZ7WOuS6Z2gmQXG6bHGUJiYmOQR4sAq8Q6ShVaQo51PgmfW9+1mLfRSYC4LyFbDtQgXKyISAHgLEYFp+4OD/yOqWrQEtqVSVDhwXZyBPlHqsmJa+v3bd44mYrfHx2OQtp9brLWvYy1EBDnSTTVxQhAASPpHDa5eeWW3mNUpNY7ohmbfxSM0HNgQ6tcnTAXV9BEa8yfM+B3JYqFsrVjt8uRn7FAL5WkhTF+8MYcgBmcihNNKwEokSLH5AL1s4wkPiIqmbq3FiBkKYFUPBM3+uZfIU8vCMVn4BRnmEgIzmI1Yu4KEtnlWa2aBOnUMehWqTMvOJwU4LAaDXQ7blNb6Y4ZVIMCkkKSMMnlVC5ImKK5xgPkzBgiJ5FJ2j20+Mm27yKbBRb2KWhRy2KYkBCU4fkZ12cmtGHe7I6pZtIIRZ8d6gsxInPSkDOQqhtLhvdz5U0Z/Y+6DiTZpEl/Ig2Ydy+8r7QKf0j1dvuRP2KqVaE9JOeSRdtmvrn5gRymCpnYg2TsHw3sqa3tjYGEJiRVlsKCgmPt0Rt3bPf8Qii9aQrMJTWTYQJ4u3GH2N2qJYDn0Vo+hRZM8BXG7+9u6SXgAGwhzWVC1VZy2ToqfHtmw9lHPjZQKpxjFMQn4wXzWGcRZugcmkp+k1zWBIZBhN9w4ZfvSvXkgN9FQGcbVo+QTDIaFFZ5vW+Lgvj6K9Fm4gXyDb65Q0HYW75EalIEQ+c0FyzwKBTCDf3owuVXiBn1oQFyg7zEKFZ4VXTZxG3Vr16gTXmVy+XyHCrJDQBGTWpemSk3LmPxRPvN44u5yRW5/oBh49kz3lNcmJ3Q1Ky0jqRXU+GGbB2Xj0OjKH3ZG9mdKEHqVE0RNUytPLwIkaOtt+cZQf0OIr5QCZ+4hBeaMFSGDvKDAyldWjJy2ZWUkxMYSWjeFg/Ai0r69HkAE4LPSWl+nL9CX8oNLfs8JmKhyWLOKkbDAs9k7o4qqd4W2nmPMoPr9j7f0OzwaU18SVtxnSPZPRGUUk/Cw5c6wWNUVHWWOsPQ1XKrN0NFSJ5TQ1fu4sLD4HiWjPeY2VHBMBUq0uxj8cqVJYb2fIq5EHpDDlCGqd7XAsKr0WIjB1o5KgqQXgahSKPe+6wz7k2qjMRk8f6LWDOtUj3ln91+aaKqn0Y2SVWLpNrRE35Ku3sPoBk5QOYzRLDLIRiYVKBhLDtRtC7tB/l1C8aBwyMF0ECrStZE4aNOjwyMnbYw5N3QPPJ7Qq3kPGX5Qd1o7a+XGOUQs8Vx8Xk+KEu8XjPcxh7S+aYQ+Qvga8P6gHvdbNtr6ESEXsSsHHhdzkNjLunGVxJwwoZbl7W1X0VBIwYMHCB2T8TgHnJsXsJ86fVk6YfpR3uGBa5AePXmzaGMfPZhNPRr6FkP5fw2+GMHGSX/RBj6cTpQTOWI69A/XwvQ0N3nf0Y2TBYhqzLzIu9t+Lu9CHUMPCeJbgGEWuUrA9PTGbqFFJbz1vCx5ynKnN7GvtpnDhtJ7kwWKd/BLPGrV0Uxo1gwJVEnthGHq+wLpzLk6cHgTLRlChXmSXIeMxdhxP6/WT3gBnIXR/YXBmcPAcWv2GfqKDTiBOQlsKkv0Kdk7bc6lgaa3UpNmjbCQ1LGfkWXAVz6X2WIvOORM3PX7hwf4RkMlAxZrIwbPxpjw2SoFC0iUa3XbD6U6CDTx46HJm9kGkbZ7fxD4iaVd+rpEF/0s6RTj0NR/2hnoHs0kVhZ9ACt7bbiaamM4SIOMCwMgVBQAGJ2BD8O7MXn1QxWoZqQ1cD9FzHJWmK5o1eP6P0rt6n9yKO7RYaVDhe5LiqOJbcnZUqqVEiQ7AkKAEFIJZbY4QHVFEnRFvQmxKMAAAVtfqUA+cm64QZ5eGl1tnFUVrBjGObaxcqPiF7vuecpKJoYZchTlgtbiFY1WWbxYeJAAa4H8Xg7DVZqYrl67Svbh3FbvXKp06vnIytSZF1c54Ds0XWeMK4AUs/6r86DvNbFw0lAeMqD67dm1z9f4BvDo1OmSfu0TIZeH3UfwyswTwo2d9yhNYYwZXW2KuwI1ZudjrUdOCltcozmv5b2dbL/ppP1kE70njWgxW4fNXBDBV1ZR8wRfrdaq2wgrFMCBpUhKwpmPsD0KUXEdMSlVf8tKYaEw8LYPI47l3wb8Y5Jht+K1mvBx5M5tzAjEhGNRiXgHi9XXqbYnxdtsgb9jl9cjW7ZN0iYZ41dWAF0vu4gH7tEQYhW2S1czc2yQh+lGkqUl6WqadW+mUdC3pUJJmS+oT6OL88Sn3QLUx27G5hsfrdNH8ZnOUl80U/dtFrcgl93XNx4C5utWFZG+xLh5ZTM/yyl3STqbz1GtHBkx6MtRH/0RNKTwZlnAzw8+Ds4HcLsBTHjkRdSTSoc5mXMXvfuqLBSUfvYSAl6zIsJWMvvVn/cjqNp4CbaGQK0TaIiU1hMKoyCBrzewaAOI2TwTVVULPpyf3NoIfO4GyD/rIP6RXhw+KkaVcPEWB4ktHtBYH6gkXY/4QpeI7HCIkr/xRgExtuDl0JrUpjMvIuafRYjLl9/Pb4xV0MH04dzNIsfLkmTyMNqoH4/T8y/i85oF30AivEYfjbEBzNKZkbYsaD2eXpS0FfvPNx3ZEKd/WtlKPZtYNWYvn3/J1PqUmz+R3FYqTYmo4f/ss8fj10bz7ObVs/kYcdAtxJnX8iP9nuq1BzJzUAuaSv3JFTDHOnAzUeM9BMM9H9Gie+OgRXHW0sOpOcSybQ20Ks4tzVXoPIxgHHrS1nb/OHbuejY3odIvUKXvQvL1yMfGPZZHvt9fjvX12xs+LDivHtNqHq/+QR3Wd6t1iDRlIcCWfjrbz1xeg+lZT9M3gNTEQ+9ssQwqyk+pzNQTNqKqVn/6t4+M2dOmRxAHHmN55t2c3SfeuS6iPVUFfQvaknstrGEl2n7QuVNIT8Hf1DDhmqIqg4lSKvdj5n1LoYsT9ERLA6qDlLIOZc3sUiVyuI5MnMvBq3E+hkoFFiWH8iRlJ2m6P4i/bKUCiFXXtwLDdZJDgHcTYi2t4MCgx+RAWlbOUF8AaFlSBkGMnP7W+DjUNLb2GM8aCfPWp6jtAVhXTZQzuWrJVeqw7Gb8ntV6FEOJi2xXbSMdCrXh4hnSEwHr9gbSVdzSfP9yqz8IP55FQk5+U+X+p+zcFkaOCOkOEtWw4U3Y8y5Ujs1f/hUcZ+8/VxgPXLc9/KIK9hfT0Dl/a0jqfCn1E98JyDHKsPU6DTxSk+GYMHVNeHPUKRc66iXUB2G7yC2dGwffsDAmmmFJXFLTydHxsqDgvu+lmWpoJB75vn9v5KYOnTe3IWNxFTPehHN+ITKFNFfmtBDP5VIYmvTtD26DAbBfrkCr9zqxC6QXL9WozXn8gu31+nPnGIf1DZ7P+9LdUWKvFh/6aJFJdMyW8nmvnYJSyOWlg/ayNzJ8trL6f8tnm6yrVg3ksebZndgLGUOrmXU4qOLO4DbIglpRRUvIpDWG0nhIrf2c9V6t3rqPaQddfCDu3oNXBcjHxC23a2oqnZFZMt/e31O2QibUXM4LlmbpHmL3KW1XN3Q7DJKsMQO0hrE3aemz3xCdt1gxkb2Fg5RSy/d5JXzsXw/pd6OflRBv/GNaZDAUPeaz/ZXxr/PrAwm7NftNlHAIBVVrOQOjTKn0zQOnVr51ejUfJBSnWKP2HJxnpU+kzJA6B/0eP6rnBXgUAmR/f4ftXPKO3WkqO3rW/6D/uDXixyIwRg3ozn8HKb8Awh6Kjm+kAngOYdVhZHzp7+y0HlWRUc3ZCdRYM3/kYOwGlxuaOp4BTWEdJr9B7AoN3nk6uiSSUzKiC7OZgcFEhgWW9wJWzYzwVU6F3UJDEJk8iJGfCI8ow3rcHJH5JNEIgnYqIdz7gZ0VvL47PXwiH3ODxDbv0eaexzMIuMl5dT3aP9aUlQHHAOe4pPbi46Tc06Ugk7p6WZ53F3XpbbOH7fkkSn98Ij5snmpm0ur6wJ3Ur9J2SC9m0W3tz8vk/yxAIbxCdcgQ6R7u0uQb3fSwx/AIxiGXCRp/Dcv5nLv29cCDNRqyY6IMtfMUmnSWiHpx+RRC2UbR9lwe2qgaTPY4R2Xp/XVBIJTSNcbHzUy8OGx9Jv+rk3FsmU5Zkmn1xFgS18U6ILE/WR/VbcUI32K2yxCHNbF40jk1uZPJOdPS6pLflCGYlPmlSmk7nGw7g4Ol95QTAwyNiXRT3MKOHJjdmu/CXz0YtqGI+OMb/uF0NnkP1dHnzJ1Ix1/fYzW6XOJEaiKM79Tk5lHvSL+wVIZlHQOZO0HRXx4Oy5S6b/l0eQcGYnz/t4g8tFgBZKJsfNPzljg+QW6l7tVjLZ/cpl1KvWL+KqoAfVjoPuVq91kHcXLBM5BrJSG1a3M/k2N4omTJxFqs+tiWuL0PsI1PdQ+BH0hpx4ILpQVrbKiNfHJzvoA6e3eLiIxPMtCNCZiGJOPzbsDkj2k6hu0CeYEjMaKjuQvTZB1I4BSbAKvWM3ZHtANG2/MUEbtjdVJNQ6Z3WsfjmWc1/fmBCdg9T3U0RR9L8Kg5bBhCsbpzqmO+UuK6TTO5iG6NzaTO2t3VSxRQZN3nQodk5RqChp2TCvzyhoMdbWLNzVQW3rNWQVv/OtL6K3nAS93SLPb2tDMhh7lTS9LGNT3sxjZY5AYAg8y7cDKLiAqFTRj2bpDv30OJ6Y+p1+6VN8P8Ms7zbaeezEXPPqGLmjI8k1FBjuLA6qODLb9TJ3p7SRuQ35IUVGnaR47B+RGcpaZe4aCMYNtczVD2qmTiCPfh7F6a0iqh7SRfgizDI+tXzyhoJsxt+1uV1l8JIvR2RqAUYiKiO6Xmu5Li/l0UQ3pQtsk07IvG2dU6WKmy5Gn8Tb7yQwAl4BZFWFZRvWb7kwaN9Ag3R6f4ynqyrL/2mBrS19JYfJgL2F/Xxz3/kPSt/1uvh/4ZVYw/XHdJRK6e4D3HHd+8ndSwbhgqKl1fk0W7vrY5QTlmvh4Nx/fKHEkgQGm1L/RYHL4/mNvq1cj3yjo9YqsmwX5Q3Fpo/KHWH2lLu/ynp8ul5UBL5/cNvxzrvnOvEi2bBVXNDTXUbr/sDXtX0oC2rkxm/Ie2kVmQ3K1dvd1RUMVF5NPN3n5z+nBR7eOMbMaEG8s2MkcqbZ4LHyNrX8s7h6m0gVGXqOPCWS4E7HJk7pkV32wHbwBG1bh8PHlEXcY+mHGsp6LmgApK35Ptm8e4ZohL38bh0L0TiP9Uew7enL4XdAdOH1pdVYJZRrYMwxbTXT0j2pyNWbtnwGPJPYXMwV9A4mSE38TsNN5b/UY6vlNLrvgNa9KD3Y/cHxk2jxNXlYtZ3ESEdkHfkcEqBDuz0c51NoHJxZ4uZ6EgRiMvL8b0pbvgf5i1ZaLe0Yx8IEZVvUnLoh2bDoUC4+Dd9UeyyppHO/cv4vKj5/vu8H/rWCTYRmoTSBYOwbYbMWibHjqfeANyx6m66mKDgvUXZW5zAaGqYQj2XvO6tc5Kt/bmKwwhHPdMlRE6iJcG0oBHIlQ8APxF+O3sO+WpEc8v1CGNrYm05LaFjzv5vbr+Yt80GGfHRKy5LqX2fERpsLv5hDubrOTl7pEJAd3yNpVvqXPgHiofBfxhlPkKSPk6BL9T6Fxa4R+1H8BGaqlAfzRxjhbjXe5P2ofI4o0hVx/0F2yxdTCJxAYhP+6kFHm+KlK0kTFEXuhW6w2zw8W8A9+0CjNfYhYZ2iZQ1pvKz8/KdXfBbwHKl3M/7JB7I6y9AuWQ5PtLyKk24BOdp7ycEaT8Ekn1wMyRlcEAG39mB/0bRHSQ1RzfzU+nYzrYsXTEdgAO6o6sJYbvbAlV5qqWNAoWJ7eW1XG5dL+lUgFYgVzXs2bJNv+vX6Jx/pzZ8fGyhl/hYH3tnB+UeiMr2Yg97EGAoSFBCmUF2cw5HHZujq3ngM8m5KFkVAPfyVL7zm5oyB1dA2X8h8kbMgLqIbD98GyuX6vm6b2HzJhbcRHGWcKTGwkB5DWk93EJ8FuJIGbrMdORmcmWlL3sxnPRZ69FvrNrMTH0C2O8U3E2pFBDvNiVabaWCwQoVP0cW0Nk8sj0LQk7wVzuz21Oul5pNBpNf+r1rjz2yppemgsb9+8aYZsIj9lUixg6Cm94pZgJ6gScPmgeFbFYO2anXGpxMr0NtTF0v1NJrJ2PqPZz9pIq4ZhvrC3nOq81GMYdpvL5l5t0knRAkEaLl0ALhmP9JvxNoIKhfgFGNhD1CzGqhaalNSaYvhXrIQM4XOqZ3mvqhzjIiI1DHq7F0Wn9SzV8nBQkg9nHFY0d+PPcRV8gtJ7VQUTAKEfdsTzlc8A18ra3mDSOQ9vl0dpaUOJ1Sw6C5l+dk51jkOHyl8ibTeZu6Sb9R/tk/kDUAA2lmbd2eTCKkqpzQpkL/dPhCo4MYh6JwM6VnWWsk2WOd8YMt5N4O4eS0PBqm0RjyLkt6u1FwoHVvvTnYEkN0J2Mxu/lOoSwQLlwJK4TO5LmJvnrXJ6qm7hrTiwMlYNUbiptkhvS97iMFgCy2QL0m4KbqIbhCxMbSOM8ZUa8xpsxhN6Y5a6XqtBX5MGpxK+bwbC7IYfHY61m0tr6Oe9vPTi5iImKfMtvVlZ9c8ALoBjWzjz9H5BzEK9ILzS2o3RtDYG56GBU7jZsO3Bw2i6nuEt1ulYQ0lY9/w+WtoSrVgNqSXl/wBVLTqk02vKzPyxSvCqaOyeCKtb0F2GRrbe2Hz8V1cTaJbVZvhafT4zsKv7xLzuXOAPPbvi95hxdhO3xV1xv7R7ugEJDL5vcZtU4JM24T0pbMPmOijw/u5f8ZNgB7MPJDQkeb0c1OhXsC5ENgGLoLfpk6xO6Ztb+2A93R3Ei/ExHxVTs7gdhUxIc3JJ8o5vnYcuvBbbE6jRF+r0JV73LUTVA5V6Wb9vvdcApOwyt9WO0M5C7cmLjz6jQeBA2oDxaNJWi3Q3Sw/NfxX8l+RTGeACBCtNMsL90ijOHr9uqGJ6Y8Tu3eRbC5csZ2ByeAgkpQ1oIFc71r1pW6CWMLrDj24P9kUn86cTK05o/5qzyv5xM4yaqn4omE3ceDHgAQH9f7rhDX/jf2y3ejaAz3cOYKLf8qoqP1JrxlfzBq+6jl/BPoFuSXX/MhL55wsRj05hBA4Q0n5Jbq2Bmo3HFHoxsBxC0Rp3kOl8OvSEqW82WvoZu4ssJoeBxrfv6lmbGfI7jrcWCb6MbcT/JVqKmX8SdkHCrN+MzebIx/m3kjqSBWE+B+Xeg68j0SYs5+l68l87xgVY54Q5SVrHNuz0b8OFq5sDGJb0L82sMBZIa1OByBB4ipKtEpd2qF1v/5jYWJR3QBfsgNXPDmsPGIhbrbGgeQFt7rA24Kf6egCf2SDkiQDfwS4WDNr91Qnca4SkvdVzkaaIC7tPLDuR7blOUcOyftOVQuHe6trHRA7fsw6mMjbf/6cYdOuQ9w==

SfYGDgceDBwMfDDJvGmoW5LUjDQskiwGngx8uPUHOAD4TBvlZOCw1n1o66cNf0Ib/1Dg1Kr6fVVdAq
wC9hll3ZKk2xr1lsW7gdcCt7T+ewHXVtVNrX8NsKh1LwJWA7Th17Xx/9g+wTR/lOSoJCuSrFi3bt3mXg5J
mtNGFhZJngKsraqVo3qOvqo6oaqWVtXShQsXTsdTStKcMX+E834c8LQkhwDbA/cA3gPslGR+23pYDFze
xr8c2ANYk2Q+cE/gml77mP40kqRpMLIti6p6Q1UtrqoldAeov1FVzwXOBJ7RRlsGfLF1L2/9tOHfqKpq7Ye
3s6X2BPYCzh5V3ZKk2xvllsVkXgecmuQ44EfAR1r7R4CPJVkFrKcLGKrq/CSnARcANwEvr6qbp79sSZq7pi
UsquqbwDdb98VMcDZTVf0OeOYk078FeMvoKpQkbYxXcEuSBhkWkqRBhoUkaZBhIUkaZFhIkgYZFpKk
QYaFJGmQYSFJGmRYSJIGGRaSpEGGhSRpkGEhSRpkWEiSBhkWkqRBhoUkaZBhIUkaZFhIkgYZFpKkQYa
FJGmQYSFJGmRYSJIGGRaSpEGGhSRpkGEhSRpkWEiSBhkWkqRBhoUkaZBhIUkaZFhIkgYZFpKkQYaFJG
mQYSFJGmRYSJIGGRaSpEEjC4sk2yc5O8lPkpyf5E2tfc8kP0iP0iyKsmnkmzX2u/S/95koNGV
bMkaWKj3LL4PXBAVT0ceARwcJL/fV/YENwJFt/COBDa39+DYeSfYGDgceDBwMfDDJvBHWLUka
Z2RhUZ0bWu+27VHAAcBnWvvJwGGt+9DWTxv+hCRp7adW1e+r6hJgFbDPqOqWJN3eSI9ZJJmX5MfA
WuAM4P8C11bVTW2UNcCi1r0IWA3Qhl8H3KvfPsE0/ec6KsmKJCvWrVs3isWRpDlrpGFRVTdX1SOAxXR
bAw8c4XOdUFVLq2rpwoULR/U0kjQnTcvZUFV1LXAm8FhgpyTz26DFwOWt+3JgD4A2/J7ANf32CaaRJE
2DUZ4NtTDJTq37rsATgQvpuMZbbRlwBdb9/LWTxv+jaqq1n54O1tqT2Av4OxR1S1Jur35w6PcYbsBJ7c
zl7YBTquqLyW5ADg1yXHAj4CPtPE/AnwsySpgPd0ZUFTV+UlOAy4AbgJeXlU3j7BuSdI4IwuLqjoXeOQE7
RczwdlMVfU74JmTzOstwFs2d42SpKnxCm5J0iDDQpI0yLCQJA2aUlgk+fpU2iRJs9NGD3An2R7YAVjQ
uAtEH3YIKrqCVJW6ehs6FeCvwdsDuwklvD4tfA+0dYlyRpFtloWFTVe4D3JPnbqnrfNNYkSZplpnSdRVW9
L8mfAUv601TVx0dUlyRpBpnSUP0i0wpLJJ8DLgf8GNg7OrpAgwLSZoDpnoF91Jg73avJknSHDPV6yx+CtxnllVIkma
vqW5Z7AJckORsup9LBaCqnjaSqiRJs8pUw+KYURYhSZrdpno21LdGXYgkafa6tlQ19Od/QSwHbAt8Juqu
seoCpMkzR5T3bLYcaw7SYBDgf1GVZQkaXbZ5LvOVucLwEEjqEeSNAtNdTfU03u929Bdd/G7kVQkSZp1p
no21FN73TcBl9LtipIkzQFTPWbxwlEXIkmavab640eLk3w+ydr2+GySxaMuTpI0O0z1APdHgeV0v2uxO/B
vrU2SNAdMNSwWVtVHq+qm9jgJWDjCuiRJs8hUw+KaJH+dZF57/DVwzSgLkyTNHlMNixcBzwKuBK4AngG8
YNNOiJqfm63Cjv3xmJn1mjY/HumbcQ3kuS+QvLSjKdo4RHX8pVCHMAvKbbA0CNVWPq3lNLtB+Wyt8+vs+tNXYOVrHqVIWvalHMNxnfBtUaeOIqOYVmzWMGNyyvSzv3dUkbL/Cn/d2JeNJwGhjxb/VQcE3UsV7TSFmWvP0ut/JeFTxBBw8+oVh3fiiJu1kkBaLb8qz5AtfI6FNbJjvnEQSTUBOSGBShJrMqMWpNW9DPhPF5xkIhNUtnsUbW4UmqbkrjzI5uf59aVLjiSxYWPLHwX6cf63DszDLhFRNDO3iBLamoXOfO9IYg5XTrtxkA0O5YOI1DMFyVWPUPfSNjEKj9rugwFYwpmoPbgx1SIMI8fL2M2PIe8NGl6kGgiAWd9CbZ4oTIgmqLPlxfGoMEOf+o4gcJ/TCrYoQAqE3q2qUCpNJEjXDVLKLrBzDiY4UjwCGD3pUVmXHzAwb/ZK5aS3ZCyW8BXURfB46mkqmQcDakRc/XdTsDFy+r6UBOm0QC3xT4YdTOzyBeBCpzEDlQJ2N/FXUMUJoFLtspd9YyVSzuDA3dvdiO6/fUeuhG2YuUomLPrbaOiF/FaOy5SyY8QsB7YtDs+a+RhI3ACzuVwp0VEFOzzRW1oxr0YN0W8Bf6FPUh4jgx9/HfLhBQiGqbhjiwhdgmHNFRw86dMm3CAUTwI1Nz2BAYQ0pmWrl8hLZ2oIiE+D6dDoSp/pNJ2t7jRR8CJ3rL4dBDN/Xr2vhGfUqqNpaV2mGvPG/cD/cMDXrmPmQr8dXFhkpQxY1gDDjf3iZWyPfR0HmIpJLLEwCIEpZvywe4t+E7r5DGB4SJA+hHRFBf84hO/xD4zMkr5eRgnKzJhsyB4DYMOLI5NA6Zk47eLvgwrpqgX3eQgGdxFBYvZAZKQyBk9Hc9l1cPtRyvdF5qn59/ZjmeyeQH//nffJ/e7+rzzgUC5d9dhR0SnJzCFnbtbTkmPsETMzxQljLa9pvXWAW3CMb4OXlBrnzBSHqCORJYHgsVCnOcwK/UH9lsd9oVPz9jTa0xm7Xvj6MYMHwmOD5ZeHAFqVJ1Uprt0wN4sL/zHQRAS0MpNhmRVtoORpfd2uS4LXX4/MJBRCiQX2q9hQOQ8MjLs5tXqGLE8sW3kMXOxV8DBGpTy0wP8mRa+8Ag6a0IjAUxc1FWYkoidL5WDkQUF4Z4SH5qdU/ccD6qA8MFIe1VSjRuKuxuLq3XsgCzg+gnvcj0vWWKL+B5uPyG3HaxDWK1sFIzrKDCSRRMT/5eCTnZs0wn9Fhm8SlycTqZ6UKcHX5K6wAjP9K5+fN1jM9iPBK+Id+7vwLXZFv0YEJUc4l6vOMaPKqujpIYIBsy3A0QTNeEVD7DK2sJ6qY4uNEVnUIv0GDZ2MFI+o8IN5pDxuCFvo+UvDUpzQhgMwAlKLgdhG9q0jxTwfKGMYOckdDeRKfpaS+l/Yc6Wl0HAvuTZu0CiH/ti/uJuyzR1eAMtDsNZOPGATJS5h1e0aNhYjV0AA9rNaCP9l9P8Oy7Vcm3aOgXGYcTqVHHAwiwGTcJsSMm5vJtaA8I7l0PZwE6hRPAuzaWomM/IxSMrz0Xiq5AJ6Qn/6Zr+DqqRzmlIWeJ5ntnH3iN5LfhsGwNQM5lsh1wMVjFNgrV8uX3xqPpAGo8X0v0pUDiUfUrd/rWI0fvKL2jiT3eL0nUnKtPlTcgcPFRO1gCPsqH3tJn6Nag7Drpo+fuEeE1Q6H7ybDWK4ucRPGLu/OkP3fGSbkrW+SCZl1BXE4Xo4/4jN4KP9yn3HTdGAfdc8XzB8DVNAsIV8wl9XIUJWAvVlxTvZAyqdZ9vsUmcDy5GW7ZzfpQHQnWmg2ASgEDUwtLTqeRQUJI6wdm5nfRI/5MdUVqwdLDTGP1d9HIIfwRkbsaRUUv0T8KBNKEFbGrLmgj2NDnJ4Ghy8Bm7F8PMDDuNkP/FfUEw3UHmOG43uzN6eNdakhTi2u9WJYBIWXK4N1QvDKPyZWZUrlbHy92ik8Dhwpeg3q7tCXpRJhgNCMZOlO35kyAXOEBuxfcUWHWrMQJv3xaqdo3cPjUG3AHIt3mfAvOeIyAyqsYEzc2NbbCP4f8lHVzGPWzuIyK3TZDyD8P96YWg/AM+l9QXq6j9DuOKVmnt2nHXD+lOaHJAzPDw2LwJvpj6Gkk/Zc3U3VH2D8n3ATE7Nn2VvSJcHTDwgo/M/fwLDkHPiGdXjvGWzQxb8YIAOcMKN+pE5GxJMZ5Jd+xBOrNfZd9Qg6sbI2jQrzmt0yBQo6UsFcgK8uCBNtxtxlHW4D3l6K2TuTGnMJdb6Kqh4mM3ZxXeUHy/kTd2NwVdpsdVUohZ4AtAE99FtUP+gj9lsfBAVwjQ5IK5UPT9yQ+6FG0W4Z2NovE0ivOE/8Gqwyz8FJ3y/9y+kI3cuwj/YeKd1XIhu/87Hs1M0xWeWfRkWDLBcUF4RRmrdUkxqJgrYzb9eHdI2I3U3VFJ0yv3ALmWP3Qam3T/

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+Aew5KleRSYGlV+ZsVM8zdUJKkQW5ZSJIGuWUhSRp
kWEiSBhkWkqRBhoV0JyS5YRPGPSbJq0c1f2mUDAtJ0iDDQtrMJrsjavPwJN9PclGSl/SmeU270++5Sd40A
2VLG2VYSJvfxu6I+jC6O6I+Fnhjkt2THEh39999gEcAj07yF9Ncs7RR3u5D2vwWA59KshuwHXBJb9gXq+p
G4MYkZ9IFxOOBA4EftXF2oAuPb09fydLGGRbS5vc+4F1VtTzJ/nT3MRoz/irYAgK8tar+ZXrKkzadu6GkzW
9jd0Q9NMn2Se4F7A+cA5wOvCjJDgBJFiW593QVK02FWxbSnXO3JGt6/e9i43dEPRc4E9gFeHNVXQFcke
RBwPeTANwAPA9YO/rypanx3lCSpEHuhpIkDTIsJEmDDAtJ0iDDQpI0yLCQJA0yLCRJgwwLSdKg/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=[EarlyStopping(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"
      ]
     },
    {
     "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"
          ]
        }
      ]
    }
  ]
}
```