

**NAME : S.MOUNIKA**  
**REG.NO.: 811519104074**

```
{
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 1,
      "id": "4200b24b",
      "metadata": {},
      "outputs": [],
      "source": [
        "import pandas as pd"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 10,
      "id": "b262c31b",
      "metadata": {},
      "outputs": [],
      "source": [
        "spam=
pd.read_csv('C:/Users/Admin/Desktop/Nalaiyathiran/assign/spam.csv',delimite
r=',',encoding='latin-1')"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 11,
      "id": "b86a1c49",
      "metadata": {},
      "outputs": [
        {
          "data": {
            "text/html": [
              "<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>"
],
"text/plain": [
" v1
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",

```

v2 Unnamed:

```

    "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 "
  ]
},
"execution_count": 11,
"metadata": {},
"output_type": "execute_result"
}
],
"source": [
  "spam.head()"
]
},
{
  "cell_type": "code",
  "execution_count": 12,
  "id": "250d57fb",
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "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",
        "---  -
        "0    v1      5572 non-null   object\n",
        "1    v2      5572 non-null   object\n",
        "dtypes: object(2)\n",
        "memory usage: 87.2+ KB\n"
      ]
    }
  ]
},
],
"source": [
  "spam.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)\n",
  "spam.info()"
]
},
{
  "cell_type": "code",
  "execution_count": 16,
  "id": "735f1e5b",
  "metadata": {},
  "outputs": [],
  "source": [
    "import numpy as np\n",
    "import matplotlib.pyplot as plt\n",
    "import seaborn as sns\n",
    "from sklearn.preprocessing import LabelEncoder"
  ]
},

```

```
{
  "cell_type": "code",
  "execution_count": 17,
  "id": "61892a0a",
  "metadata": {},
  "outputs": [
    {
      "name": "stderr",
      "output_type": "stream",
      "text": [
        "C:\\Users\\Admin\\anaconda3\\lib\\site-
packages\\seaborn\\_decorators.py:36: 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",
        "    warnings.warn(\\n"
      ]
    },
    {
      "data": {
        "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAYsAAAEWCAYAAACXGLsWAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG90bGliIHZlcnNpb24zLjQuMywgHR0cHM6Ly9tYXRwbG90bGliLm9yZy/MnkTPAAACXBIWXMAAAsTAAALEwEAmpwYAAAZwkLEQVR4nO3de7RdZX3u8e9DQECBAiUgJGioxVbAKzFitZV6I9VaGO3B4pEaKxrLodWeYVWw5yhqGdLq0apVWnoxQas01VrTC7WIYusRiaFeIiAlA5DERBKQqxe040/8Md+UyWbvPXcga++d7O9njDXWnO+c71zvnGvt9az5zstOVSFJ0mR2m+kGSJmP8NCKjTIsJAKDTIsJEmDDatJ0iDDQpIOyLDQlCVZkeQPZuilk+RDSW5Nsmac6S9P8oWZaNu010T4JBtnuh3SWIbFTizJD UluSvKIXtkrklw6g80alWcCzwMWvtWsmW6MNNcYFju/3YHXznQjtleSedtZ5dHADVX1vVG0R9LkDIud3zuB30uy/9gJSRYlqSS798ouTfLKNvzyJP83yXuS3JbkuiQ/18o3JNmSZNmYxR6U50Ikdyb5fJfJH95b9s23ad5Nck+TfVwkrkpyX5J+TfA/4xXHaeliSla3++iSvauWnAX8BPD3JXUneOthGSPKu1lVlfZJf6pX/ZpKrW7uvS/Lq3rTjk2xM8oa2zpuTnJTBUn+s7XnTZ085guTfCXJHW27nT3Oe7AsyY1Jbk7y+73pe7dtc2uSq4CnTvI6ae/VliS3J/16kmN62/dPJ3lv3tvadkeSK5L8fG/a2Un+NslHWT11SR6b5Kz2WhuSPH+Sdt2Q5PwtPd9L8pdJDKlyUVveZ5Ic0Jv/uCrfbJ+5ryU5vjft5e39ubO9hy9t5T/dlun2tg3/ZorrtneSlW37Xt3e44296Ycl+USSre3lXtObtiTJ2rbcm5K8e6JtMCdUly+d9AHcADwX+DvgDlrZK4FL2/AioIDde3UuBV7Zh18O3AP8JjAP+APgRuADwJ7A84E7gX3a/Cva+C06e8FvtCmpQLY0Ja10/AU4Gbg6F7d24Fn0PlI2Wuc9fk88EFgI+BjWfbgOb22fmGSbFfy4EfAq9q6nA5sAtKmvxB4DBDgWcd3gae0ace37fBmYI+2jK3AR4F9gaOBHwI/NcFrHw88vq3XE4CbgJPGvAd/DuwnPBG4G3hcm34u80/AgcDhwDeAjrR08zgnAFcD+bt0eBxw69N606accCP9nem9cB39n2HgBnt/U7oU2/ALge+P3e9rh+4HP4JeaQYAGwBfgP4MmtLZ8F3tLmXQDcArygba/ntfH5dJ+h04CfafMeyn2fn4+19uzWPh/PnOK6nUv3uToAWAh8fdv2bcu6or3vDwn+Crg0OKFNvwz4jTa8D3DcTP/Nz+j3zUw3wMdDePPuC4tj6L6I57P9YXFtb9rj2/yH9MpuAZ7UhlcAF/am7QPcS/cl9+vAv49p35/1viRWABdMsi6Ht2Xt2yt7B7Ci19ahsFjfgG394W5dHTjD/3wOvbcPHAZ8A5rXxfVvdp/Xmv4IWAfN4X/4YeM+Y92Bhb/oa4JQ2fB2wtDdtOROHxbOB/wSOA3YbM23C92aCzd0KPLENnw1c3Jv2IuCucbbH/pN8D1/ag/8EcF5v/HeAv2/DbwQ+PKb+p4fldGFxG/BrwN5j5rkAOL+/HSfZ/v11+68v/zB+Su4Li6cBN46pexbwoTb8b8BbgYMeyt/prvKwG2oXUFXfAP4ROPNBVL+pN/yDtryXZfv0xjfoXvcu4LvAYXTHFJ7WuhZuS3Ib8FLgkePVHcdhwHer6s5e2bfofol01Xd6bft+G9wHIMkvJfLS61K6je6X7UG9urdU1blt+AftebLt8F+SPC3J5lpXxu3Ab41Z9v3aRrdXs21zh3H/7fKtiVauqj4L/Andnt9NSc5Psl9v1oneG5K8rnXD3N7W/yfGtHHsut48zvYYd/OnqD/Rtns0cPKYz8kz6faQvfkf3o+O3gM1J/inJz7Z6b6Dbmlqt5Mokr9i28IF1G7t9+8OPBg4b05Y30e0hAZwGPBb4ZpIvJ/n1SdZ/12dY7DreQtdd0P9y3XYw+OG9sv6X94Nx+LaBJPvQdZ9sovsj/HxV7d977FNVp/fqTnaL403AgUn27ZU9Cvj2Q2wvSfak+7X7Lrq9pv2Bf6b78tkRPGqspvsV/xPAn27HsjfT26Z06zyhqnpfVR1L1zX2WOD1vcnjvjetD/+NwIuBA9r6374dbdyRNTDtWfQ/J4+oqnmBqurTVfU8ui6ob9J131FV36mqV1XVYcCrgQ+24xhD67aZrvtpm/623kDXvdZvy75V9YL2mtDW1UuAg4E/BD6e3pmHc41hsYuoqvXA3wCv6ZVtPufyPTXJvPzr7DEP8aVekOSZSR4GvB24vKo2003ZPDbJbyTZoz2emuRxU2z/BuCLwDus7JXkcXS/7P76IbYXuv7oPemOQ9yT7sD3hAdsH4R96faKfphkCfDft6PuKuCsJACKWUjXZTOutj2f1mQPuh8CP6TratpmovdmX7pjMluB3ZO8GdiPmfER4EVJTMi fbyb3SnWCwsB0U/5X2hXw3XVfyvQBjTm7b7pupmrThtatv30XAL/dm7YGuCPJG9uB8HlJjknlyPaapyaZX1U/puseg/tv7znFsNi1vI2u73bfvXS/Pm+h+zX6Yf4Gh+124v5LnASXVcTrfvo+CapdHsJ36H7Nbbndj7JXR9/JuAT9I8D774IbZ3W9teQ/fFcSvd1/nqh7rcnv8BvC3JnXQHS1dtR9230nU9XQ/8K/DhSebdj+6X9q2tzil0e0vbjPve0B0TuIjueMe36EJmsi7BkWNhdSjdd8/W1o7X030XU7Z3gHoT3To8i27bQneW2OVJ7gJ7715bVdczvG5vAzbsbd/PAB+nCyJan9uL6E6muJ7uhIy/oOvGAlqKXNle8710x51+uMM2xk5
```

m25kiknZiSVbQHbj9XzPdltksyel0X/rPmum27Gzcs5C0y0pyaJJnJNktyc/Q7b18cqbbtTPafX  
gWSdppPYzuFO4j6I47XEh3LY+2k91QkqRBdkNJkgaNtBsQyQl0tyC4F7inqhYnOZDuFM9FdFd+v  
riqbm3zn0V3uus9wGuq6tOt/Fi6K1T3pjs//rUlsEt00EEH1aJFi3b40knSruyKK664uarmjy2f  
jmMWv1hVN/fGzwQuqapzk5zZxt+Y5Ci60y6Pprvq8jNJHttObzuP7jYIX6ILi6V0p8tNaNGiRax  
du3bHr40k7cKSjHsXgZnohjoRWNmGVWIn9covrKq72/nT64ElSQ4F9quqy9rexAW9OpKkaTDqsC  
jgX9ttg5e3skOqajNAez64ls/g/hfTbGxlC9rw2PIHSLK83VJ47datW3fgakjS3DbqgbqhnVNWmJ  
AcDFyf55iTzjnefmpqk/IGFVefT3ZmSxYsXe5qXJO0gI92zqKpN7XkL3YUwS+julnkodBFM0N37  
Hro9hv5NvhbSXfa/kfvfCGxbuSRpmowsLJI8YtsdRNUwZ5P949dVtPdu572/Kk2vBo4JcmeSY4  
AjpgTWtK6qO9P9d60AL+vVksSRNg1F2Qx0CfLL7fmd34KNV9S9JvgysSvevMm8ETgaoqiuTrAKuor  
uL5Bm9++mfzn2nz17EwJlQkqQda5e9gnvx4sXlqbOStH2SXFFVi8eWewW3JGmQYSFJGuRdZydw7  
OsvmOkmaBa64p0vm+kmSDPCPQtJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMIwksYMM  
C0nSIMNckjTIsJAKDTIsJEmDDatJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMIwksSYM  
MC0nSIMNckjTIsJAKDTIsJEmDDatJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNGnlyJJ  
mX5CtJ/rGNH5jk4iTxtucDevOelWR9kmuSnNarPzbJujbtfUky6nZLku4zHXsWrWu7o2fCVxSV  
UcCl7RxxkhFnAicDSwFPphkXqtzHrAcOLi9lk5DuyVJzUjDIslC4IXAX/SKTwrWtuGVwEm98gur  
6u6quh5YDyxJciiwX1VdVlUFXNcrI0maBqPes/hj4A3Aj3tlhlTVZoD2fHArXwBs6M23sZUtaMN  
jyx8gyfIka5Os3bp16w5ZAUnSCMMiyS8DW6rqiqlWGaesJil/YGHV+VWluKoWz58/f4ovK0kasv  
sIl/0M4FeSvADYc9gvyUeAm5IcWlWbWxftljB/RuDwXv2FwKZWvnCccknSNBnZnkVVnVVVC6tqe  
d2B689W1anAamBzm20Z8Kk2vBo4JcmeSY6gO5C9pnVV3ZnkuHYW1Mt6dSRJ02CUexYTORdyleQ0  
4EbgZICqujLJKuAq4B7gjKq6t9U5HVgB7A1c1B6SpGkyLWFRVZcCl7bhW4DnTDDfOca545SvBY4  
ZXQs1sZPxcm5J0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMIwksYMMc0nSIMNckjTIsJ  
AkDTIsJEmDDatJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMIwksYMMc0nSIMNckjTIs  
JAKDTIsJEmDDatJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMIwksYMMc0nSIMNckjRo  
ZGGRZK8ka5J8LcmVsd7ayg9McnGsa9vzAb06ZyVZn+SaJcf0yo9NsQ5Nel+SjKrdkqQHGuWexd3  
As6vqicCTgKVJjgPOBC6pqiOBS9o4SY4CTgGOBpYCH0wry3rPGA5cGR7LB1huyVJY4wsLKpzVx  
vdoz0KOBfY2cpXAie14ROBC6vq7qq6HlgPLElyKLBfVv1WVQVc0KsjsZoGiZlmkWRekq8CW4CLq  
+py4JCq2gzQng9usy8ANvsqb2xlc9rw2PlXm95krVJl7dunWHroskzWUjDYuquireqngQspNtL  
OGAs2cc7DlGTlI/3eudXleKqWjx//vztbq8kaXzTcjZUVd0GXEp3rOGmlrVEe97SztsIHN6rthD  
Y1MoXjlmUSzomzwan6S/dvw3sBzgWk84TrkbaZlWkfa8GrglCR7JjmC7K0D2mtZvDwEs49pZUC  
/r1ZEktYPdR7jsQ4GV7Yym3YBVVfWPSS4DViU5DbgROBmgqg5Msqg4CrgHOKQ7m3LOh1YAewNX  
NQekqRpMrKwqKqvA08ep/wW4DkTlDkHOGec8rXAZMc7JEkj5BXckqRBhoUkaZBhIUkaNKWwSHLJ  
VMokSbumSQ9wJ9kLeDhwULvh37YL5PYDDhtx2yRJs8TQ2VCvBn6XLhiu4L6wuAP4wOiaJUmatSY  
Ni6p6L/DeJL9Tve+fpjzJkmaZKV1nUVXvT/JzwKJ+naq6YETtkiTNiLMkiyQfBh4dfBXYdlX1tt  
uFS5J2cVO9gnsxcFT7fxKSpDlmgtdZfAN45CgbIkmaavaa6Z3EQcFWSNXT/LhWAqvqVkbRkKjSrT  
DUszh5lIyRJs9tUz4b6/KgbIkmaavaZ6NtSd3PevTB8G7AF8r6r2G1XDJEmzx1T3LPbtjyc5CVgy  
igZJkmafB3XX2ar6e+DZO7YpkqTzaqrduL/ag92N7roLr7mQpDliqmdDvag3fA9wA3DiDm+NJG1  
Wmuoxi98cdUMkSbPXVP/50cIkn0yyJclNST6RZOGoGydJmh2meoD7Q8Bquv9rsQD4h1YmSZoDph  
oW86vqQ1V1T3usAOaPsF2SpFlkqmFxc5JTk8xrj1OBW0bZMEs7DHVsHgF8GLgO8Bm4L8BHvSWp  
DliqqfOvh1YVlW3AiQ5EHgXXYhIknZxU92zeMK2oACoqu8CTx5NkyRJs81Uw2K3JAdsG217FlPd  
K5Ek7eSm+oX/f4AvJvk43W0+XgycM7JWSZJmlalewX1BkrV0Nw8M8KtVddVIWyzJmJWm3JXUwsG  
AkKQ56EHdolySNLcYfPkQYaFJGnQyMIiyeFJPpfk6iRXJnltKz8wycVJrm3P/VNyz0qyPsk1SU  
7olr+bZF2b9r4kGVW7JUKPNMo9i3uA1lXV44DjgDOSHAwCCVxSVUcCl7Rx2rRTgKOBpcAHk8xry  
zoPWA4c2R5LR9huSdIYIwLqtpcvf/Rhu8Erqa7vfmJwMo220rgpDZ8InBhVdlDvdcD64ElSQ4F  
9quqy6qqgAt6dSRJ02BajlkkWUR3e5DLgUOqajN0gQIc3GZbAGzoVdvYyha04bHl47308iRrk6z  
dunXrDl0HSZrLRh4WSfyBPgH8blXdmms45TVJOUPLKw6v6oWV9Xi+fp9dxuStKOMNCyS7EEFXH  
9dVX/Xim9qXUu05y2tfCNweK/6QmBTk184TrkbaZlWkmyoAH8JXF1V7+5NWG0sa8PLGE/1yk9Js  
meSI+gOZK9pXVV3JjmuLfnlvTqSpGkwyjvHPgP4DWBdkq+2sjcB5wKrkwG3AicDFBVyZZRXdl  
kXuAM6rq3lbvdGAFsDdwUXtIkqbJyMKiqr7A+McbAJ4zQZ1zG0dutlW1Fjhmx7VokrQ9vIJbkjT  
IsJAKDTIsJEmDDatJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMIwksYMMc0nSIMNckj  
TIsJAKDTIsJEmDDatJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMIwksYMMc0nSIMNck  
jTIsJAKDTIsJEmDDatJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNGllyJpMrJFuSfKNX  
dmCSi5Nc254P6E07K8n6JNckOaFXfmySdW3a+5JkVG2WJl1vlHsWK4ClY8rOBC6pqiOBS9o4SY4  
CTgGOBnU+mGReq3MesBw4sj3GLlOSNGIjC4uq+jfgu2OKTwRWtuGVwEm98gur6u6quh5YDyxJci  
iwX1VdVlUFXNcrI0maJtN9zOKQqtoM0J4PbuULgA29+Ta2sgVteGz5uJiSt7I2ydtW7fu0IZL0  
lw2Ww5wj3ccoiYpHldVnV9Vi6tq8fz583dY4yRprpvusLipdS3Rnre08o3A4b35FgKbWvnCccol  
SdNouSniNbCsDS8DPtUrPyXJnkmoDuQvaZ1Vd2Z5Lh2FtTLenUkSdNk91EtOMnHgOOBg5JsBN4  
CnAusSnIacCNmKbVXZlkFXAVcA9wRlXd2xZl0t2ZVXsDF7WHJGkajSwsquole0x6zgTznwOcm0  
75WuCYHdg0SdJ2mi0HuCVJs5hhIUkaZFhIkgYZFpKkQYaFJGnQyM6GkjQ6N77t8TPdBm1Cj3rzu  
pEt2z0LSdIgw0KSNMiwkCQNMIwksYMMc0nSIMNckjTIsJAKDTIsJEmDDatJ0iDDQpI0yLCQJA0y  
LCRJgwwLSdIgw0KSNMiwkCQNMIwksYMMc0nSIMNckjTIsJAKDTIsJEmDDatJ0iDDQpI0yLCQJA0  
yLCRJgwwLSdIgw0KSNMiwkCQNMIwksYNN2mrBIsjTJNUnWJz1zptsjSXPJThEWSeYBHwB+CTgKee

```
mSo2a2VZI0d+wUYQEsAdZX1XVV9f+AC4ETZ7hNkjRn7D7TDZiiBcCG3vhG4G1jZ0qyHFjeRu9Kc
s00tG0uOAi4eaYbMRvkXctmugl6ID+f27w1O2Ipjx6vcGcJi/G2QD2goOp84PzRN2duSbK2qhbP
dDuk8fj5nB47SzURuDW3vhCYNMMtUWS5pydJSy+DBYz5IqkDwNOAVbPcJskac7YKbqhquqeJL8
NfBqYB/xVVV05w82aS+za02zm53MapOoBXf+SJN3PztINJUmaQYaFJGmQYTGhJVmU5Bsz3Q5Js5
9hIUkaZFhoXpI/T3Jlkn9NsneSVyX5cpKvJfLEkocDJFmR5Lwkn0tyXZJnJfmrJFcnWTHD66FdQ
JJHJPmn9tn7RpJfT3JDkj9MsqY9frrN+6Iklyf5SpLPJDmk1Z+dZGX7PN+Q5FeT/FGSdUn+Jcke
M7uWOyFDQkcCH6iqo4HbgF8D/q6qnlpVTwSuBk7rzX8A8GzgfwL/ALwHOBp4fJInTW07tWtaCmy
qqidW1THAv7TyO6pqCfAnwB+3si8Ax1XVn+nuF/eG3nIeA7yQ7h5yHwE+V1WPB37QyrWdDATdX1
VfbcNXAIuAY5L8e5J1wEvpwmCbf6jufOtlwElVta6qfgxc2epKD8U64LltT+Lnq+r2Vv6x3vPT2
/BC4NPtc/p67v85vaiqftSWN4/7Qmcdfk4fFMNCd/eG76W7UHMf8Nvtl9hbg3Gmf/HY+r+mJ3k
Ik/NX1X1n8CxdF/q70jy5m2T+rO15/cDf9I+p69mnM9p+yHzo7rvgjI/pw+SYaHx7Atsbn27L53
pxmjushIY8P2q+gJwLuApbdKv954va8M/AXy7DXs74BEzYTWe/wlcDnyL7hfevjPbHM0hjwfeme
THwI+A04GPA3smuZzuB+5L2rxnA3+b5NvAl4Ajpr+5c4e3+5A0qyW5AVhcVf7Pihlkn5QkaZB7F
pKkQe5ZSJIGGRaSpEGGhSRpkGEhPURJ7tqOec9O8nujWr40KoaFJGmQYSGNwER3RG2emOSzSa5N
8qpende3u/1+PclbZ6DZ0oQMC2k0Jrsj6hPo7nz6dODNSQ5L8ny6OwAvAZ4EHJvkF6a3ydLEvN2
HNBoLgb9JcijwMOD63rRPVdUPgB8k+RxdQDwTeD7wlTbPPnTh8W/Tl2RpYoafNBrvB95dVauTHE
93H6Ntxl4JW0CAD1TVn01L66TtZDeUNBqT3RH1xCR7JflJ4Hjgy8CngVck2QcgyYIkB09XY6Uh7
llID93Dk2zsjb+bye+Iugb4J+BRwNurahOwKcnjgMuSANwFnApsGX3zpWHeG0qSNMhuKENsSIMNC
kjtIsJAKDTIsJEmDDAtJ0iDDQpI0yLCQJA36/x/W84YoIJYvAAAAAE1FTkSuQmCC\n",
```

```
    "text/plain": [
      "<Figure size 432x288 with 1 Axes>"
    ],
    "metadata": {
      "needs_background": "light"
    },
    "output_type": "display_data"
  },
  "source": [
    "sns.countplot(spam.v1)\n",
    "plt.xlabel('Label')\n",
    "plt.title('Number of ham and spam messages')\n",
    "X = spam.v2\n",
    "Y = spam.v1\n",
    "le = LabelEncoder()\n",
    "Y = le.fit_transform(Y)\n",
    "Y = Y.reshape(-1,1)"
  ],
  {
    "cell_type": "code",
    "execution_count": 21,
    "id": "f5ce9bec",
    "metadata": {},
    "outputs": [],
    "source": [
      "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.preprocessing.text import Tokenizer\n",
      "from keras.preprocessing import sequence\n",
      "from keras.callbacks import EarlyStopping\n",
      "%matplotlib inline"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 22,
```

```

    "id": "7fa642b8",
    "metadata": {},
    "outputs": [],
    "source": [
        "X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15) "
    ]
},
{
    "cell_type": "code",
    "execution_count": 25,
    "id": "d7cb7862",
    "metadata": {},
    "outputs": [],
    "source": [
        "from keras.preprocessing.sequence import pad_sequences"
    ]
},
{
    "cell_type": "code",
    "execution_count": 26,
    "id": "a7cc04ed",
    "metadata": {},
    "outputs": [],
    "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)"
    ]
},
{
    "cell_type": "code",
    "execution_count": 27,
    "id": "da3dd18f",
    "metadata": {},
    "outputs": [],
    "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"
    ]
},
{
    "cell_type": "code",
    "execution_count": 28,
    "id": "84a6807b",
    "metadata": {},
    "outputs": [
        {
            "name": "stdout",
            "output_type": "stream",

```

```

    "text": [
        "Model: \"model\\\"\\n\",
    ]
}

"=====\\n",
" Layer (type)                Output Shape                Param #
\\n",
" inputs (InputLayer)         [(None, 150)]               0
\\n",
" embedding (Embedding)       (None, 150, 50)            50000
\\n",
" lstm (LSTM)                  (None, 64)                  29440
\\n",
" FC1 (Dense)                  (None, 256)                 16640
\\n",
" activation (Activation)      (None, 256)                 0
\\n",
" dropout (Dropout)           (None, 256)                 0
\\n",
" out_layer (Dense)           (None, 1)                   257
\\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",
    "
    ]
}
],
"source": [
    "model = RNN()\\n",
    "model.summary()"
]
},
{
    "cell_type": "code",
    "execution_count": 31,
    "id": "08345681",
    "metadata": {},
    "outputs": [],
    "source": [

```



```

    "from tensorflow.keras.optimizers import RMSprop"
  ]
},
{
  "cell_type": "code",
  "execution_count": 32,
  "id": "4f099089",
  "metadata": {},
  "outputs": [],
  "source": [

"model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['acc
uracy'])"
  ]
},
{
  "cell_type": "code",
  "execution_count": 33,
  "id": "f4948eb2",
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "Epoch 1/10\n",
        "30/30 [=====] - 6s 144ms/step - loss:
0.3212 - accuracy: 0.8743 - val_loss: 0.1448 - val_accuracy: 0.9473\n",
        "Epoch 2/10\n",
        "30/30 [=====] - 4s 129ms/step - loss:
0.0865 - accuracy: 0.9810 - val_loss: 0.0528 - val_accuracy: 0.9873\n"
      ]
    },
    {
      "data": {
        "text/plain": [
          "<keras.callbacks.History at 0x207c54e3a90>"
        ]
      },
      "execution_count": 33,
      "metadata": {},
      "output_type": "execute_result"
    }
  ],
  "source": [
    "model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,\n",
    "
validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',min_delta=
0.0001)])"
  ]
},
{
  "cell_type": "code",
  "execution_count": 34,
  "id": "b7d49ac1",
  "metadata": {},
  "outputs": [],
  "source": [
    "model.save('Spam.h5')"
  ]
},

```

```

{
  "cell_type": "code",
  "execution_count": 35,
  "id": "293cd4b2",
  "metadata": {},
  "outputs": [
    {
      "data": {
        "text/plain": [
          "array([[ 0,    0,    0, ..., 386, 696, 100],\n",
          "       [ 0,    0,    0, ..., 82, 259,  2],\n",
          "       [ 0,    0,    0, ..., 296,  27, 338],\n",
          "       ..., \n",
          "       [ 0,    0,    0, ..., 621, 377, 190],\n",
          "       [ 0,    0,    0, ..., 93, 143, 11],\n",
          "       [ 0,    0,    0, ..., 408, 744, 480]])"
        ]
      },
      "execution_count": 35,
      "metadata": {},
      "output_type": "execute_result"
    }
  ],
  "source": [
    "test_sequences = tok.texts_to_sequences(X_test)\n",
    "test_sequences_matrix =\n",
    "pad_sequences(test_sequences,maxlen=max_len)\n",
    "test_sequences_matrix"
  ]
},
{
  "cell_type": "code",
  "execution_count": 36,
  "id": "265b0358",
  "metadata": {},
  "outputs": [
    {
      "name": "stdout",
      "output_type": "stream",
      "text": [
        "27/27 [=====] - 0s 13ms/step - loss: 0.0951\n",
        "- accuracy: 0.9749\n",
        "Accuracy: 0.9748803973197937\n",
        "Loss: 0.09510093927383423\n"
      ]
    }
  ],
  "source": [
    "accr = model.evaluate(test_sequences_matrix,Y_test)\n",
    "print('Accuracy:',accr[1])\n",
    "print('Loss:',accr[0])"
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
  "id": "e027ce8c",
  "metadata": {},
  "outputs": [],
  "source": []
}

```

```
],
"metadata": {
  "kernel_spec": {
    "display_name": "Python 3 (ipykernel)",
    "language": "python",
    "name": "python3"
  },
  "language_info": {
    "codemirror_mode": {
      "name": "ipython",
      "version": 3
    },
    "file_extension": ".py",
    "mimetype": "text/x-python",
    "name": "python",
    "nbconvert_exporter": "python",
    "pygments_lexer": "ipython3",
    "version": "3.9.7"
  }
},
"nbformat": 4,
"nbformat_minor": 5
}
```