NAME : G.ROOPASHRI

REG.NO.: 811519104090


```
{
 "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                                                      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  "
      ]
     },
     "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",
      "---  ------  --------------  ----- \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":
"iVBORw0KGgoAAAANSUhEUgAAAYsAAAEWCAYAAACXGLsWAAAAOXRFWHRTb2Z0d2FyZQBNYXRwbG
90bGliIHZlcnNpb24zLjQuMywgaHR0cHM6Ly9tYXRwbG90bGliLm9yZy/MnkTPAAAACXBIWXMAA
AsTAAALEwEAmpwYAAAZwklEQVR4nO3de7RdZX3u8e9DQECBAiUgJGioxVbAKzFitZV6I9VaGO3B
4pEaKxrLodWeYVVWw5yhqGdLq0apVWnoxQas01VrTC7WIYusRiaFeIiAlA5DERBKQqxeO4/8Md+
UyWbvPXcga++d7O9njXWnO+c71zvnGvt9az5zstOVSFJ0mR2m+kGSJJmP8NCkjTIsJAkDTIsJE
mDDAtJ0iDDQpI0yLDQlCVZkeQPZui1k+RDSW5Nsmac6S9S9P8oWZaNuOlOlOT4JBtnuh3SWIbFTizJD
UluSvKIXtkrk1w6w80alWcCLwDm19WiMe1G0bkVjTd3fjqusJZOGWNPbu+wfLXeNUX27n9q0JTRV9pYvVG0GR9Lk
DIud3zuB30uy/9gJSRYlqSS798ouuTfLKNvzyzvzJP83vmDP3bwYXhOkQ19Ts84tWrxnGyqMWmsb3aWm/
yIaW5/34yh3u/KEB2sqJMaTy7MZI7xKiHYVNk2sDfNA2ykQgVxAz6bDsp/BkY3Y4G46yHlGq8aXjLQX3Wdyb1hnpF3rIWAlsa7lqsZIJuW4lmJXCi7bw39R93jPf3dW9FRlNjAM2FmSeR9zGcUPUaT+x0nrpgW9FuJ9ZPDlebAMZ0KSXk96Hq3QoWbQTFdYhknSDVtSQHtuj4/0ObZuhbnyh9pdO8Zw9+pfLrOQGqSC2Ba7HwQ2DpOuC+A9ZvJoz/D19Qm9CQwCmGsQtqmr+BS5XlH/x0j8vp6M0/9SozivHYtUBmmnkD2Ns9tMcQSo6cbA7p6n6pVnYMmtrcBm8eZXzcLjzsfScxw1D6cnmi8Rt33m3ZM8qrbJ9XKZk0KfbaivfrcYVs6wOrkK+QDCpL4l4JCBV1nf/ca+7nhw8oYEr0bvo+h3H1MM+8iUQYMNwRrZCYnWaR1rE6KiFu9gn6dc82l28A8s8aG1a8ubnrzlmrD+9Q6Lv1QYmFUxyTsWvKG+hMpvFu4eUGc3BvLy10KbVUm6yJIiJNJ6T0Nh4DDHn5xe/Ctl1MKB5jlBAqQoIJapSXjGMpDVwfTjMg7RE9R3yx3Zgh7+vOs3VuRcdBOZ4GwkyDWZvUCGsEGYVS1HWSuMAh9Lc0FhJ/sSNiSDQALEY6F0lWaqG0tFLyqYFg/2sd7zqtyXz2J1ru1vkyNnzZbYd9z/YUnDK4Dm6Ij3uWn+QOCJ9kTcDYRdTbABs3ABhThuNtDsCAvU/rr+RdMOm0KPN3pqcnVjAyQrtN2b2VlVP3S7ajz0/jbFH1AlkdNcWBZRFFzDZeNawOEOlwvvYyFKD3fSnfs6WRgB8FF/5KKmlzZq4CzgNCZ13IQFa61KlDnnd4pSn/MlHHvPQKmxDXHzBlPYSy4Jqrj/Gjq0tqOVMNAXE2yrftGdXTm/2N8K3aaxR3oY2Zl32yYxUpIp7tNmKiH4Ah4YfeKwEr9bETjUgqGfmUmiGQy6JxIXDSpoH7jyRUK3Qn/vXaKhfmmxDqrjDXAd/xDYWI9pb0q8tfXlbpWF2HJFK1Z9a4bKJ7kHtJ2/p7qD2TaoTeYMgIiwdz3ltaSrTA7fa9vQUhSlP48t7JdQaLaGnu+D+yzYT8vLX/xQvRQYgLQyxk2ulbXtrkF2YE8h5n6ZPBapYzs5VCUB6+qiWvXwRLk4sHXn8PGdWVMzoUlt2JzERc5XngBZTCbr5Yr/Xj94xTGpbeOJZK8dUDM/ZG5T9osZbjeY0qX6lSeA04m6X9t2XKXH+ke3fWcjmaBDN4N24T+6tBK68/HfPQiNW1fE3gDU+FIWhLx2u3DgQzoivJtTAQTw5YyeIYBOcGZeLOJAZT2QRbhZmjW6Sng0hGAaJTk6NTWF6OqZgkEnIVDvhLy43uXnbvmkhD8HWqzeS9DHyjL8Eovzlab+Zky4xK5zsLy0OqEVnpH/rc4mOHeuYPoSeaMnb3KXW2H2zz2OQ59iFgYpVZ0UFQnVtcoDrBDTA2MHLkXdnGn+C88yzyPsinu74svqb5EmhavIwojM5MHBeE1H5ldSHWv08axFG0DD5FI1IVvqBbE8EkAmoB8/QmGnCLW
uS7JXkCXS/7P76IbYXuv7oPemOQ9yT7sD3hAdsH4R96faKfphkCfDft6PuKuCsJAckWUjXZTOut
j2flmQPuh8CP6TratpmovdmX7pjMluB3ZO8GdiPmfER4EVJTmifyb/b/YX/5X2hXw3XVfY
vQBJTm7bB7pupmrThtatv30XAL/dm7YGuCPJG9uB8Hl1Jjkny7JXR9/JuAT9
Id77j4IbZ3W9teQ/fFcSvdl/nqh7rcnv8BvC3JnXQHS1dtR9230nU9XQ/8K/DhSebdj+6X9q2tz
i10e0vbjPve0B0TuIjueMe36EJmsi7BkWnhdSJdd8/W1o7X030X7UZ3gHoT3To8i27QbQneW2OVJ
7qJ7715bVdczvG5vAzbSbd/PAB+nCyJaN9uL6uOvGAlgKXNle8710x5l+uMM2xk5

m25kiknZiSVbQHbj9XzPdltksyel0X/rPmum27Gzcs5C0y0pyaJJnJNktyc/Q7bl8cqbbtTPafX
gWSdppPYzuFO4j6I47XEh3LY+2k91QkqRBdkNJkgaNtBsqyQ10tyC4F7inqhYnOZDuFM9FdFd+v
riqbm3zn0V3uuS9wGuq6tOt/Fi6K1T3pjs//rU1sEt00EEH1aJFi3b4OknSruyKK664uarmjy2f
jmMWv1hVN/fGzwQuqapzk5zZxt+Y5Ci60y6Pprvq8jNJHttObzuP7jYIX6ILi6V0p8tNaNGiRax
du3bHr40k7cKSjHsXgZnohjoRWNmGVwIn9covrKq72/nT64ElSQ4F9quqy9rexAW9OpKkaTDqsC
jgX9ttg5e3skOqajNAez64lS/g/hfTbGxlC9rw2PIHSLK83VJ47datW3fgakjS3DbqbqhnVNWmJ
AcDFyf55iTzjnefmpqk/IGFVefT3ZmSxYsXe5qXJO0gI92zqKpN7XkL3YUwS+julnkodBfM0N37
Hro9hv5NvhbSXfa/kfvfCGxbuSRpmowsLJI8YtsdRNuNwZ5P949dVtPdu572/Kk2vBo4JcmeSY4
AjgTWtK6qO9P9d60AL+vVkSRNg1F2Qx0CfLL7fmd34KNV9S9JvgysSvevMm8ETgaoqiuTrAKuor
uL5Bm9++mfzn2nzl7EwJlQkqQda5e9gnvx4sXlqbOStH2SXFFVi8eWewW3JGmQYSFGuRdZydw7
OsvmOkmaBa64p0vm+kmSDPCPQtJ0iDDQpI0yLCQJ0yLCRJgwwLSdIgw0KSNMiwkCQNMiwkSYMM
C0nSIMNCkjTIsJAkDTIsJEmDDAtJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMiwkSYM
MC0nSIMNCkjTIsJAkDTIsJEmDDAtJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNGnlYJJ
mX5CtJ/rGNH5jk4iTXtucDevOelWR9kmuSnNArPzbJujbtfUky6nZLku4zHXsWrwwWu7o2fCVxSV
UcCl7RxkhwFnAIcDSwFPphkXqtzHrAcOLI9lk5DuyVJPz/SKTwRWtuGVwEm98gur
6u6quh5YDyxJciDIslC4IXAX/SKTwRWtuGVwEm98gur
6u6quh5YDyxJciiwX1VdVlUFXNCrI0maBqPes/hj4A3Aj4A3Aj3tlh1TVZoD2fHAr7mfd34KNV9S9JvgysSvevMm8ETgaoqiuTrAKuor
jyx8gyfIka5Os3bp16w5ZAUnSCMMiyS8DW6rqiqlWGaKsJil/YGHV+VW1uKoWz58/f4ovK0K0kasv
sIl/0M4FeSvADYC9gvyUeAm5IcWlWbWxfTljb/RuDwXv2FwnCcknSNVnVnVVC6tqE
d2B689W1anAamBZm20Z8Kk2vBo4JcmeSY6gO5CdRdYleQ0Q0
4EbgZICqujLJKuAq4B7gjKq6t9U5Hd8AxCuTh8YGBrY2K8JgrVfUZmHKJxSWcZyqm00OcD8f4VhlEzT
ZXQslSZPxCm5J0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMiwkSYMMC0nSIMNCkjTIsJ
AkDTIsJEmDDAtJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMiwkSYMMC0nSIMNCkjTIs
JAkDTIsJEmDDAtJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMiwkSYMMC0nSIMNCkjRo
ZGGRZK8ka5J8LcmVSd7ayg9McnGSa9vzvzAb06ZyVVZn+SaJCf0yo9Nsq5Ne1+SjKrdkqQHGuWexd3
As6vqicCTgKVJgOBC6pqiOBS9o4SY4CTgGOBpYCH0wyry3rPGA5cR7LB1huyVJY4wsLKpzVx
vdoz0KOBFY2cpXAie14ROBC6vq6qq6HlgPLElyKLBfVV1WVQVc0KsjSZoGIz1mkWekq8CW4CLq
+py4JCq2gzQng9usy8ANvSqb2xlC9rw2PLxXm95krVJ1m7dunWHroskzWUjuquereqngQspNtL
OGaS2cc7DlGTlI/3eudX1eKqWjx//vRzbq8kaXzTcjZUVd0eXEp3rOG3m3r4/BXJPqxHIV
Y1MoXjlMuSZomozwbanCS/dvwH/9Y/5JhTJ1Gq5M7t5vpgsUdgGCvxIcfG+BlDcdjZ1u9Wc6+ZC3j/
/r1ZEkTYPdR7jsQ4GV7Ym3YBVVfWPSS4DViUUjdbgROBmgqq5Msb5Mrgq4Msgq4CrgHOKOq7m3LOh1YAewcLX
NQekqRpMrKwqKqvA08ep/wW4DkTxkT1DkHOGec8rXAZMc7JEkj5BXhoUkaZNhHLJ
VMokSbumSQ9wJ9kLeDhwULvL0vn0LvWnFmc+ieDyVnFBa9SiXwBR5XLhiu4L6wuAP4wOiaJUmaTSY
Ni6p6L/DeJL9TVe+fpjZJkmaYKZ0NVVXvT/LzwIJ+naq6YETtkiTNIFMKiyR/CRwJfBW4txXfd18mSdJuMAxhIX
uFS5J2cVO9gSeWIQ9tVAY6G9ayvQd7wDIxxyX5DX5sXw7yKPn7x7o3JB4SDsR4PCAmqvBnGbkRKkjSrT
DUszh5SUzJHIQIuDM8E2C/0sP01u1TtI4+4K6UYBsKm2KsTyRzcDZb8GfXbyeyrPZq6uyZs2
igzJKmafB3XX2ar6eDZO7YpkqTZaqrdU/aG92N7roLr7mQpDliqmdDvag3fA9wA3DiDm+NJG
WmuoxU98cdUMkSbPXVP/50cIkn0zn3RByETeMoD7Q8Bquv9v6rsQ4h1YmSz0Dph
oW86vqQ1V1T3usAOaPsF2bFlkqmFxc5JTK8xrj7xDHVsHgF8GLgO8Bm4L8BHvsWp
DliqqfOvh1YVlW3AiQ5EHgXXkYXhIknZxU92zeMK2KoACoqu8CTx5NkyRJO81Uw+JB7d9aAvfrj3h/PuW5
K5Ek7eSm+oX/j+4Avvw1WdFl0gacmjuubzhkY4Blvuv4vH4O3wbDjbm4GPAaUiUGJiVaYpaqTdc7Yv3
AkKQ56EHdolySNLcYFKkKQYaFIGnTQYMIiyeyFJPpfk4fk68yVrG6DGTDwgpmjafJp3/WrT0wx3pB2Ggpvq5
7olR+bZF2b9r4kGVW7JUKzNqMo9i3uA11xV44pDjgSGOBpcAHk8xSVVdUdcD64EljRwcFj0gx+c2Ei8NY
zoPWA4c2R5LptA7vfmWehldpOSwFX8OgoEM9a5kvdbcrqqG8Ti6O+ubQEG0BiQe2c5Xy4b6oR0B6z
dunXrDl0HSZrLh4WSfYBPgH8mYGDpHHaXsf3XdmG6Dm5oqV6RHkWYP8JXd5xA40WqK5MCCyCyS7ZFFXDH0XFH
9dVX/Xim9qXUu05y2tfCNweeK/6QmBTK184TrkaZQM8myoAH8JXF1V7+5NWg0sa8PLgE/1yk9JsmeSI4AjgTWtq
meSI+gOZK9pXVV3JjmuFNlTqSpGkwyjvHPgXkwwG7fwB08DWkDBv8C6GdXhuX8mXL7G2WckhpkDgvDbf0E4GkGJ2iLuvatzd
kXuAM6rq3lbvVGAFsDwvbFGuWKqbJyMKiqr7A4XfGkXWiJ+QFKqt7A+McbAJ4z1ZlzGOdutlW1Fjhmx7VOkrQ9vIJbkjT
IsJAkDTIsJEmDDAtJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMiwkSYMMC0nSIMNCkj
TIsJAkDTIsJEmDDAtJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNMiwkSYMMC0nSIMNCk
jTIsJAkDTIsJEmDDAtJ0iDDQpI0yLCQJA0yLCRJgwwLSdIgw0KSNMiwkCQNGllYJPmrJFuSfKNX
dmCSi5Nc254P6eOK8n6JNckOaFXmmySyfd3SaUS5NWvhOlCkBePPAoFHKTbmCkSmCdW3a+5JkVG2WJI1vlHsWrYqyxYslAXmYpGx1Y4S4gV4T7
CTgGObnU+mGReq3MesBw5sj2WTAkj+jfgu2OKTKTwRWtuGVwEm98gur6u6quh5YDyxJciDyxJci
iwX1VdVlUFXNCrI0maJtN9zOKQqtoM0J0M0J4PbuULgA29+Ta2sgVteGz5uJIsT7I2ydqtW7fu0IZL0
lw2Ww5wj3ccoiYpf2Bh1flVtbiqFs+fP3+HNk6S5rLpDoub28XtaFS7uw9tnKc4rxxfbNnZVnnDjzzDx1KYXsV+
SdNousNiNbCsDS8DPtUrPyXJnkmOAjuQBsVbvUfhNvD4xp/1PnPbiNoDnTmYzTzH8G+uQPM4NnS8JN5tG8eXb4
CnAusSnIacCNwMkBVXZlkFXAVcA9wRlXd25Z1OrAC2Bu4qD0kSdNkpGFRVa9YCBpz3PpBMdwtnHf76s0t8BB
75WuCYHdg0SdJ2i0HuCVJs5hhhIUkaZFhIkgZ0m6d0rvcYVGfZaFkKQYaFFxZlG6LsmJ2oN+trXtKM6lsZp5KfQYkMsFQ
pEt2z0LSdIgw0KSNMiwkCQNMiwkSYMMC0nSIMNCkjTIsJAkDTIsJEmDDAtJ0iDDQpI0yLCQJA0y
LCRJgwwLSdIgw0KSNMiwkCQNMiwkSYMMC0nSIMNCkjTIsJAkDTIsJEmDDAtJ0iDDQpI0yLCQJA0
yLCRJgwwLSdIgw0KSNMiwkCQNMiwkSYN2mrBIsjTjTJNUnWJzlzPWY3xZzlptsjSXPJThEWSeiEcB2ScBkHwB+CTgKg
lJScBZwIkiAFB2WFxtaXHHD0BdYHEQTXsLfILBZyeJSQGlL
X6EXMKbBGZyvWC8Syjl4vuxcSi5c6ShrDxUgBMaHDeBdIzgTcHk4AwJrpVeFvMqsK+jHhLJjOrtjVzJ5m1
mnZMa4FZJy5sZGzFDbqzLe4ERAUwkHMt1IR6VjV3Xwz/vqWfKi6HWg2h35DgAhSe9ZNlPOD1nrk
c5gS2KJt6xSGlCnrLNS1Kf8zoNWyI+ZyrrwE88eAT9yk3Qd8e7XiE9uZ1M5tR7fpX2VGXrJCJkhImD4kg
tmZHqJ+ToNs4z6PnsO9v18N4WkTcfvb/Yq4ETjLkHjuWBAhrfV1Q3yVGP8AdsJmHyE

mSo2a2VZI0d+wUYQEsAdZX1XVV9f+AC4ETZ7hNkjRn7D7TDZiiBcCG3vhG4GljZ0qyHFjeRu9Kc
s00tG0uOAi4eaYbMRvkXctmugl6ID+f27wlO2Ipjx6vcGcJi/G2QD2goOp84PzRN2duSbK2qhbP
dDuk8fj5nB47SzfURuDw3vhCYNMMtUWS5pydJSy+DByZ5IgkDwNOAVbPcJskac7YKbqhquqeJL8
NfBqYB/xVVV05w82aS+za02zm53MapOoBXf+SJN3PztINJUmaQYaFJGmQYTGHJVmU5Bsz3Q5Js5
9hIUkaZFhoXpI/T3Jlkn9NsneSVyX5cpKvJflEkocDJFmR5Lwkn0tyXZJnJfmrJFcnWTHD66FdQ
JJHJPmn9tn7RpJfT3JDkj9MsqY9frrN+6Iklyf5SpLPJDmklZ+dZGX7PN+Q5FeT/FGSdUn+Jcke
M7uWOyfDQkcCH6iqo4HbgF8D/q6qnlpVTSuBk7rzX8A8GzgfwL/ALwHOBp4fJInTWO7tWtaCmy
qqidW1THAv7TyO6pqCfAnwB+3si8Ax1XVk+nuF/eG3nIeA7h5yHwE+V1WPB37QyrWdDAtdX1
VfbcNXAIuAY5L8e5J1wEvpwmCbf6jufOt1wE1Vta6qfgxc2epKD8U64LltT+Lnq+r2Vv6x3vTPT2
/BC4NPtc/p67v85vaiqftSWN4/7Qmcdfk4fFMNCd/eG76W7UHMF8Nvtl9hbgb3Gmf/HY/+r+mJ3k
Ik/NXlX1n8CxdF/q70jy5m2T+rO15/cDf9I+p69mnM9p9p+yHzo7rvgjI/pw+SYaHx7Atsbn27L53
pxmjuSHIY8P2q+gjwLuApbdKv954va8M/AXy7DXs74BEzYTWe/w1cDnyL7hfevjPbHM0hjwfeme
THwI+A04GPA3smuZzZuB+5L2rxnA3+b5NvAl4Ajpr+5c4e3+5A0qyW5AVhcVbdOuQdL4Ajpr+5c4e3+5A0qyW5AVhcVf7PihlkN5QkaZB7F
pKkQe5ZSJIGGRaSpEGGhSRpkGEhPURJ7tqOec9Oct9OIunc1d0O6nujWr40KoaFJGmQYSHNgCTPT3J
8qpende3u/1+PclbZ6DZ0oQMC2k0Jrsj6hPo7nz6dOODNSQ5L8ny6OwAvAZ4EHJEvN2
HNBoLgb9JcijwwMOD63rRPVdUPgB8k+RxdQDwTeD7wlTbPPnTh8W/T12RpoaFNBrvAfavqv2d/dVaaTeB
93H6Ntxl4JW0CAd1TVn01L66TtZDeUNBqT3RH1xCR7JfJjHgy8gkB0cgyIkB09XY6Uh7
llID93Dk2zsjb+bye+Iugb4J+BRwNurahOwKcnjgMuSAnwFnApsGX3zpWHeG0G0qSNMhuKEnSIMNC
kjTIsJAkDTIsJEmDDAtJ0iDDQpI0yLCQJA36/x/W84YoIJYvAAAAAElFTkSuQmCC\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",
"=================================================================\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"
     ]
    }
   ],
   "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 =
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
- 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": {
  "kernelspec": {
   "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
}
```