```
{
  "nbformat": 4,
  "nbformat_minor": 0,
  "metadata": {
    "colab": {
      "provenance": []
    },
    "kernelspec": {
      "name": "python3",
      "display_name": "Python 3"
    },
    "language_info": {
      "name": "python"
    }
  },
  "cells": [
    {
      "cell_type": "markdown",
      "source": [
        "##**Download the dataset**"
      ],
      "metadata": {
        "id": "s_ic6MDnjlML"
      }
    },
    {
      "cell_type": "code",
      "execution_count": 1,
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "Nf3hUzCFO78r",
        "outputId": "f01d2311-cac4-4a12-d888-1894730fffb4"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Mounted at /content/drive\n"
          ]
        }
      ],
      "source": [
        "from google.colab import drive\n",
        "drive.mount('/content/drive')\n"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "##**Import the required library**"
      ],
      "metadata": {
        "id": "YKkZVFlTjrra"
      }
    },
```

```
    {
      "cell_type": "code",
      "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 tensorflow.keras.models import Model\n",
        "from tensorflow.keras.layers import LSTM,Activation, Dense,
Dropout, Input,Embedding\n",
        "from tensorflow.keras.optimizers import RMSprop\n",
        "from tensorflow.keras.preprocessing.text import Tokenizer\n",
        "from tensorflow.keras.preprocessing import sequence\n",
        "from tensorflow.keras.utils import to_categorical\n",
        "from tensorflow.keras.callbacks import EarlyStopping"
      ],
      "metadata": {
        "id": "bZENJh0UPqnM"
      },
      "execution_count": 12,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "###**Read the dataset**"
      ],
      "metadata": {
        "id": "U0pvoaxYj0qE"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "import csv\n",
        "with open('/content/drive/MyDrive/Colab Notebooks/spam.csv',
'r') as csvfile:\n",
        "  reader = csv.reader(csvfile)"
      ],
      "metadata": {
        "id": "-Nc9b-DYPFK_"
      },
      "execution_count": 8,
      "outputs": []
    },
    {
      "cell_type": "code",
      "source": [
        "df = pd.read_csv(r'/content/drive/MyDrive/Colab
Notebooks/spam.csv',encoding='latin-1')\n",
        "df"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 1914
```

```
        },
        "id": "8PWps2LLPpm_",
        "outputId": "a6b4832a-c850-47d8-b30f-796bb9602bc4"
      },
      "execution_count": 15,
      "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",
              "5567  spam  This is the 2nd time we have tried 2 contact
u...        NaN   \n",
              "5568   ham              Will Ì_ b going to esplanade fr
home?        NaN   \n",
              "5569   ham  Pity, * was in mood for that. So...any other
s...        NaN   \n",
              "5570   ham  The guy did some bitching but I acted like
i'd...        NaN   \n",
              "5571   ham                         Rofl. Its true to its
name        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  \n",
              "...         ...        ...  \n",
              "5567        NaN        NaN  \n",
              "5568        NaN        NaN  \n",
              "5569        NaN        NaN  \n",
              "5570        NaN        NaN  \n",
              "5571        NaN        NaN  \n",
              "\n",
              "[5572 rows x 5 columns]"
            ],
            "text/html": [
              "\n",
              "  <div id=\"df-7793bfb6-b702-4827-b115-d6d96cfda229\">\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",
               "    <tr>\n",
               "      <th>...</th>\n",
               "      <td>...</td>\n",
               "      <td>...</td>\n",
               "      <td>...</td>\n",
               "      <td>...</td>\n",
               "      <td>...</td>\n",
               "    </tr>\n",
               "    <tr>\n",
               "      <th>5567</th>\n",
               "      <td>spam</td>\n",
               "      <td>This is the 2nd time we have tried 2 contact
u...</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
               "    </tr>\n",
               "    <tr>\n",
               "      <th>5568</th>\n",
               "      <td>ham</td>\n",
               "      <td>Will Ì_ b going to esplanade fr home?</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
               "    </tr>\n",
               "    <tr>\n",
               "      <th>5569</th>\n",
               "      <td>ham</td>\n",
               "      <td>Pity, * was in mood for that. So...any other
s...</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
               "    </tr>\n",
               "    <tr>\n",
               "      <th>5570</th>\n",
               "      <td>ham</td>\n",
               "      <td>The guy did some bitching but I acted like
i'd...</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
               "    </tr>\n",
               "    <tr>\n",
               "      <th>5571</th>\n",
               "      <td>ham</td>\n",
               "      <td>Rofl. Its true to its name</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
               "      <td>NaN</td>\n",
```

```
              "    </tr>\n",
              "  </tbody>\n",
              "</table>\n",
              "<p>5572 rows × 5 columns</p>\n",
              "</div>\n",
              "      <button class=\"colab-df-convert\"
onclick=\"convertToInteractive('df-7793bfb6-b702-4827-b115-
d6d96cfda229')\"\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-7793bfb6-b702-4827-
b115-d6d96cfda229 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-
7793bfb6-b702-4827-b115-d6d96cfda229');\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": 15
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "##**Pre processing**"
```

```
      ],
      "metadata": {
        "id": "V8U-LKPMj62i"
      }
    },
    {
      "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": "h1OHRZiPQSCd",
        "outputId": "88998c13-ca52-4b42-c0a9-a5bad14c1549"
      },
      "execution_count": 16,
      "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": "markdown",
      "source": [
        "######**Count the spam and ham**"
      ],
      "metadata": {
        "id": "Mad9SMBLk_0V"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "sns.countplot(df.v1)\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 352
        },
        "id": "oa4TBq1wQtfM",
```

```
        "outputId": "de262b95-8ad6-4214-e94e-26823d8af913"
      },
      "execution_count": 17,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stderr",
          "text": [
            "/usr/local/lib/python3.7/dist-
packages/seaborn/_decorators.py:43: FutureWarning: Pass the following
variable as a keyword arg: x. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without
an explicit keyword will result in an error or misinterpretation.\n",
            "  FutureWarning\n"
          ]
        },
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "<matplotlib.axes._subplots.AxesSubplot at 0x7f475b28f590>"
            ]
          },
          "metadata": {},
          "execution_count": 17
        },
        {
          "output_type": "display_data",
          "data": {
            "text/plain": [
              "<Figure size 432x288 with 1 Axes>"
            ],
            "image/png":
"iVBORw0KGgoAAAANSUhEUgAAAYsAAAEHCAYAAABfkmooAAAABHNCSVQICAgIfAhkiAAAAAlw
SFlzAAALEgAACxIB0t1+/AAAADh0RVh0U29mdHdhcmUAbWF0cGxvdGxpYiB2ZXJzaW9uMy4yL
jIsIGh0dHA6Ly9tYXRwbG90bGliLm9yZy+WH4yJAAARFElEQVR4nO3de7BdZXnH8e+PBLwVJU
pETGjDaGY6IPXSU8DaTi2MgFqFsYg4KtEyjdPBjnZaFTutKMMVlvE6wwWJGgr4K1Ea8UUsNU
ZBRJRw6WUFKEk5RJNQK2VGnj6x36jm3AO74FmnXOS8/3M7NlrPetdaz97Zs/5nbX2WmunqpAk
6aHsMdsNSJLmPsNCktRlWEiSugwLSVKXYSFJ6jIsJEldC4fceJBgB8B9wHbqmAfUW2qbTCEFd
+2bNCwaH63qr4/Nn8qcFlVvTvJqW3+rcALgeXtcRjwFi6nARNAAeuSrK6qe9u8fwlcxuj49r3
7du0w70aSdlNJbp1q2WwchAkXMy1bm+Ky1J89nQ
YVHAV5KsS7Ky1farqtvb9B3Afm16CXDb2LobW22q+gMkWZlkbZK1mzdvnpQVmD1ZQPQ/1WV
W1K8mRgTZJ/G19YVZVkp9xvpKrOAc4Bmji8B4mkrQTDbpnUVb2vNdwOeBQ4E72+El2bbQv
gm4ICx1Ze22lR1SdIMGSwskjwuyd47poEjgWuA1cCOK5pWAF9q06uBE9rVUIcDW9phqguBI5Nsm
yqG3n0qH6liQ92JCHofYDPj86I5aFwN9X1ZeTXA1cnORk4FbghDb/BFcn8xOm93A6NTZ1wFU1ZYk
7wKubuNOr6otA/YtSdpBdsdblE9MTJbnmspayau4N4l/fqbL
5jtFjQHrXvvSbPdgjQr3LOQJHUZFpKkLsNCktRlWEiSugwLSVKXYSFJ6jIsJEldhoUkqcuwkC
R1GRaSpC7DQpLUZVhIkroMC0lSl2EhSeoyLCRJXYaFJKnLsJAkdRkWkqQuw0KS1GVYSJK6DAt
JUpdhIUnqMiwkSV2GhSSpy7CQJHUZFpKkLsNCktRlWEiSugwLSVKXYSFJ6jIsJEldg4dFkgVJ
rknyxTZ/YJIrk2xIcl2SvVr9UW1+Q1u+dGwb72z1G5IcNVY/utU2JDllqNcgSZrbTDJqF9ajUmb
7ANqqGSsXNGtrMVpYcm2EjpGnLDOaxPUpmZrNy1yzjgqVndgzsNjlYq6ZTFjhn3Kaxx9vvBN6Y
6pKa9E0OY6Wo3qS7zx4JDBeyFlv7yPKdvD3XzymVSqU/2aWLgQ2zbo6O2fhPcUVt0jOfsUKE2bsk
2+bSX/WtvW7/vb6KZ8wyd7BjLpoSy1zm7KAGu2UOQH5CMWvGZt/n9KBCoKVEd6ZNZIsSf6I
1XzDwuWrUVBnI8UzGzAFuXD27WtrNOrJfPdVo5NaXqnE8KUaEJyqhR7Z/KF+mB1XfXpGSa5JAea
KvXBB6wAf6ZWSpCufLzvf26RQvz1ihlHqTpG6eU1DRd8Ocna+EhVvZW0cjFfR9vPLhMfR8cUX
qq1yUnjKs3aAWz3PXDu3hS6Dsx6sOm+kw8Ml1J5oK8fAI3/uOVc6TLBYzuY/Ikg4tOYkWk51yO
jVw4fOHmEJOOV4bzfa5XOH91a4EpDtPkbV4+kjv4Uy4ZRGlcbQU8kT1AobZRzXP8zqH4VlyMhj6
JL1eLU1B4LlNw7J7BLq1d9VTqDd9hypOtVrK3dG6j3N6JrGzLEO9OzDXdrkWMdn/+EYZbD2QF5pz
C2EzBjxxWN/fo8dt3Xut5i6tKdn0WSDjQtg+8jUbWYe+9Drhd/QSz0Y+/ATwdWndAfy5adT+i
GDe8CozdYkVdBw0Ljl2JH9pmbq5P8t1oY2UjyVVFhcyTJJJRHE8tXYuaCXV3Xk+aGhCQbHYuLRXz
cL3KWcXbvZ3JMLIpehLaETTb+5DtfQdwLDr0dXCMZPuJ/4jDjBIvbZL3yIBW+PuCFxqv/Oz
l5tp1eD5t8AIQ4e/O8OqM5PYp+0YFR7I5/XbNR1RHYLVUrhHhOe9W5eD+VzqTHpSEGJwN7tjj7
6IVtdnPgdHeLUDtWGU8F2iCgBS1D5o5GS+BIBH4Mp30Io2GuPtQkyb0mhs1ySDXsQBk+Mwf
TM7vTnA5AiNS6hEjW5ZV5AHHagkOc0SDW8fb4gjjoPR0K1nErwdk2FoFJaDnH6+NgUp0uyvQ
JdVHb5YZeGPddwj2f+NRCsL3Q9lH5UftHA0m3DubIJ6Z4CHL4NUEHqEDuEnTxEg2xVcgXWNkGhJ
f27ikpazmsj8HLYgnGzlw5rXBwz7aj+hnlfMJyLXRbhHBpqf4EqXuBuPqRONIujZnwBPrb8bNk
MHFDSjbO+fbiC/GiPpaxHyaZJ8n9Dh3uNU9VkMq6BlxKJ0yN4gHXiO28udnqnPgcOO4fH99HmOzI
p/8Xd5oWbU
```

C0lSl2EhSeoyLCRJXYaFJKnLsJAkdRkWkqQuw0KS1GVYSJK6DAtJUpdhIUnqMiwkSV2GhSSpy
7CQJHUZFpKkLsNCktRlWEiSugwLSVKXYSFJ6jIsJEldhoUkqcuwkCR1GRaSpC7DQpLUZVhIkr
oMC0lSl2EhSeoaLCySPDrJVUm+k+S6JO9s9QOTXJlkQ5KLkuzV6o9q8xva8mVj23pbq9+Y5Oi
hepYkTW7IPYt7gSOq6pnAs4BjkhwOvAc4q6qeDmwFTm7jTwa2tvpZbRxJDgJOBA4GjgE+kmTB
gH1LknYwWFjUyI/b7J7tVdXAsDwAcAm4BwAWJFkpvk++SpAS1ivmsQ4n7c6BhAtUCQIlB3SHSbMzbdWhxpVGhq+VEm3Bq5J8dcmhIk3GSDRUUF2uGGAHu4
Uxbj2F8j2A3YXnnQYRgWcAkuLd/8WgTffthuBRhsXk6m2uXQ8Bt2Mt9klJTZ2jqh43FwEXz2/u
tdMUnF7cDvzHtSHXmnD7x0SIaNU5L1xSkcBOLKEgqIBPVhJgVpRaDN6u3nQRdhTKhdvG3J+RW
Fpvoej7dCOaq7oWmQUKaJ/lykehU6KUkRjF6UFsSYGq65NMqupe7kDzHyBDIvpBWsZ+MB1dmxL
9KEP+ZhWO7CfM/1vgkDR4zaqk3SdLiy2MXKRMX+qjt2iqoc7o5HgiN/M7XLsUoDPK4/43G0Xq
rMkNAbiN9+/3d6FnW6E4A/MvzRc4c8Re5GNqzCCuaPGDgZ3/ijqaJNfcVRexhqZDtlXqjFg3lM
GNqm/O2fLQeQR+2nvnfTpfrN0ZbMs73DSRgxQh5UW/wzSNi1IMNaC67LjRmBD3NFa5R4MYSS
6Hb2rzP6+zYJmmcC3ZXswdtvknwFn4q1ck8YA9gmkBJbOGaQZRdfM1GMHZGK/W0vtqQkOk5n
XOEz3RQdKU11r39/3sVjw4B8AjD/ZE3bDGFMNF0jKKyOz2wf+4iCfsO+J9uhyqeC5gEaHZddd
ebWxnOUZ0kaOH8OFOySG/NBf4r5sCgAVTF8AOqZCTHfsNjRB/f0wJ4/V9lvtaoeI22vzT6M8
p5X9jUhZf2nNwKV4zO9xWSI1Qpfq2l2HbndxN+HS5OOdYa5JdXT9VKH7ze3/ZaP0FgCXIx8p
Ov6Hr/lxbqrCvUsEbl5RefmVwhRgKYXTGmgvOAe27H4wdO8/12oxdrqjgHgDMjn9OCBrh0ihHc
3ZH6mUFFkfnzDSaS7L2v1Ld2MbafBfMS4Z6fUk7WJMDOEhSxG8lS2WLCFKtRH9Zh20LTsJ3Q
6ni7SXCaNMLoG2KgHO4EBMlhPpPs0c2lOMTAW2hqRSMi3UGHqkfW8mSyCS+kD79OsJ30FMf0M6
HAZSxqT+Qk6KIMb/8QqsJsfJBo9ZP1EPQtYlZs1+4eMdu7g80q3p5sC2lhFKPRzhGyhUWcOAJ
sNeNTFVq8WDlFwKh1nLEm3SF7z4y9qB+9S7FrAnAL7CTGWsvhq8K6EXEBQdrHY/CqKGmuzPTp
oYhTezpeetKUoSVFVtJAZX6LvDItlYHqQvLsLbAbXqWIDm9/dExuGMGqZPRCGYyW+RcdjJPTSy
tDwKhIjSzvAgsI21ocyLGVIakJ8B9PHl3HGqkzRWXNRoT8LxNgUE5fglYa7lpsNZyOS1Qth7kc
CR1GRaSpC7DQpLUZVhIkroMC0lSl2EhSeoyLCRJXYaFJKnLsJAkdRkWkqQuw0KS1GVYSJK6DAt
JUpdhIUnqMiwkSV2GhSSpy7CQJHUZFpKkLsNCktRlWEiSugwLSVKXYSFJ6jIsJEldhoUkqcuw
kCR1GRaSpC7DQpLUZVhIkroMC0lSl2EhSeoyLCRJXYaFJKnLsJAkdRkWkqQuw0KS1GVYSJK6
DAtJUpdhIUnqMiwkSV2GhSSpy7CQJHUZFpKkLsNCktRlWEiSugwLSVKXYSFJ6jIsJEldhoUk
qcuwkCR1GRaSpC7DQpLUZVhIkroMC0lSl2EhSeoyLCRJXYaFJKnLsJAkdRkWkqQuw0KS1GVYSJK6DAtJUpdhIUnqMiwkSV2GhSSpy7CQJHUNFhZJzktyV5Jrx2pPTTVeGrJ8kHkmxI8t0kzxlbZ0Ubf1OSFUP
1K0ma2pB7FucDx+xQW4W4rKqWA5e1eYAXAsvbYyXwURiFC3AacBhwKHDa9oCRJM2cwcKiqv4V
2LJD+VhgTbu/Bjhuov7xGrkCWJDkIcDRwCVq3AIDtFcgbgFIHvZAKHqpu8VKNX2K8F9qtDsHZ
u6n2K1/MS7Lf4yYQ3Sjn8FLHWqNzLhRCqdsqOS6BPEtJpvttl0X1Zd3qswVrnMNIH0h8Qu6hyM
D4+AkNvb5eF35S/jjMd5s+1pRMyWs5LjFtz4m7Hg9kkaB4eFvxt7XMRQMm+wPP3VbmxTEyszR+
v0Nlx2YcQ4L5jQxlWC3X6S6ZaTb2EGcrT3eH2Srcu2a27CFFLSC6kPo/HhXhp+VRG+IpFgFvSu
JTd1vmLCgT1DYrq0cRBvS8zsdQLBrOFO5ORZQ5Kvxd/Iw9VaJvh5Xp6NkA8zqrtpR/Oi0mtHH4C
JIBgxYAw3PJa+4T8g2qZfGpB29UyXFvhhTX6U1SO07gWWdcTjXtgyiSrm7V1qm6E2SVMA2F/cd
L2+P/uVQYETAeVJgdGl9+pQXSoaIFUbTFgKH+iM5hLfZLnimmXOWZe0Mf0TfHzuhfl1n1+3Ybf/2xBR2MzmLTTgKwG5gQJpJhgV16x2V+SCb1YRQQ4JX01cJjsvNPkdSsUbFJHa9
B5DSAjLS1qunXpNE20NoPzQ4Pz4mJTuo6cgUgXiLy2zIW/CrzIh20i7aeJUL0aD4VxzeuxDJMY9VV7KE6lTyKKxrJ2Y8ZyKqg+BcuFjmJNPBTQSVUKz9YKDiBPeWQ6zMB2yzRa6mL5tmMFaPqVHJkOsBAl5ELLVGbgpm7ZeExExOy0Cmx9Ogcr4xF2qE7GxBO7RzMCe2hqBxFKhS60DG9r6F8XEdzTRlGQ5jnuO
GNqA5mt0nJiMx9r3dJx3L4KvtMWR0mvZcXMdRgrsCSDb3gAUxhAgKAuFQqWTUUEJn8+ImVrgoKnMCKN1VmZFhlsvbGBEeAWL4b5xUsQmjNyMq/7m/Nb0dAzVDMOL6y1C+OVQY/+T8zuZUaeYMrxsjR6vJ/Xvp1zVO0fCN2zUiPjp0oXN0SkgCtk2eYX+1Bf6j6N4oerGYBvbVpszi8i4d6dJ5vm/OJBwGG2AF+IM1JSgNEKQU9RVYmJqODvnjXgE7F2Rh7N/RxrnrVs9ClFJFq4fSlZXfqGW3vfKr0OSsqFN4PoFqQDjg==\n"

```
          },
          "metadata": {
            "needs_background": "light"
          }
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [
        "X = df.v2\n",
        "Y = df.v1\n",
        "le = LabelEncoder()\n",
        "Y = le.fit_transform(Y)\n",
        "Y = Y.reshape(-1,1)\n",
        "X_train,X_test,Y_train,Y_test =
train_test_split(X,Y,test_size=0.20)\n",
        "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 =
sequence.pad_sequences(sequences,maxlen=max_len)"
      ],
      "metadata": {
        "id": "vrzHnT1pQ8Nf"
      },
      "execution_count": 38,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "##**Create Model and add Layers (LSTM,Dense-(Hidden Layers),
Output)**"
      ],
      "metadata": {
        "id": "IbVdA-GOjJ7Q"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "def RNN():\n",
        "  inputs = Input(name='inputs',shape=[max_len])\n",
        "  layer =
Embedding(max_words,50,input_length=max_len)(inputs)\n",
        "  layer = LSTM(128)(layer)\n",
        "  layer = Dense(256,name='FC1')(layer)\n",
        "  layer = Activation('relu')(layer)\n",
        "  layer = Dropout(0.5)(layer)\n",
        "  layer = Dense(1,name='out_layer')(layer)\n",
        "  layer = Activation('tanh')(layer)\n",
        "  model = Model(inputs=inputs,outputs=layer)\n",
        "  return model\n"
      ],
      "metadata": {
```

```
        "id": "8KBci7F3jV-O"
      },
      "execution_count": 48,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "###**Compile the Model**"
      ],
      "metadata": {
        "id": "fayzclNKVDa7"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model = RNN()\n",
        "model.summary()\n",

"model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['a
ccuracy','mse','mae'])"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "-AJAj4T8VB5e",
        "outputId": "1984ef38-ecab-4585-93ec-bbd1bd2e180d"
      },
      "execution_count": 39,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Model: \"model_2\"\n",

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

"=================================================================\n",
            " inputs (InputLayer)         [(None, 150)]             0
\n",
            "
\n",
            " embedding_2 (Embedding)     (None, 150, 50)           50000
\n",
            "
\n",
            " lstm_2 (LSTM)               (None, 128)               91648
\n",
            "
\n",
            " FC1 (Dense)                 (None, 256)               33024
\n",
```

```
            "
\n",
            " activation_4 (Activation)   (None, 256)               0
\n",
            "
\n",
            " dropout_2 (Dropout)         (None, 256)               0
\n",
            "
\n",
            " out_layer (Dense)           (None, 1)                 257
\n",
            "
\n",
            " activation_5 (Activation)   (None, 1)                 0
\n",
            "
\n",

"=================================================================\n",
            "Total params: 174,929\n",
            "Trainable params: 174,929\n",
            "Non-trainable params: 0\n",

"_________________________________________________________________\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "###**Fit the Model**"
      ],
      "metadata": {
        "id": "3y2ZuDwjUZQY"
      }
    },
    {
      "cell_type": "code",
      "source": [

"model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,validation_s
plit=0.2)\n",
        "\n"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "wFM7XkEaSRi0",
        "outputId": "ccd3fad9-1011-44bc-e88d-079ede852d77"
      },
      "execution_count": 46,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
```

```
          "text": [
            "Epoch 1/10\n",
            "28/28 [==============================] - 16s 568ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n",
            "Epoch 2/10\n",
            "28/28 [==============================] - 27s 967ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n",
            "Epoch 3/10\n",
            "28/28 [==============================] - 15s 536ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n",
            "Epoch 4/10\n",
            "28/28 [==============================] - 15s 533ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n",
            "Epoch 5/10\n",
            "28/28 [==============================] - 15s 536ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n",
            "Epoch 6/10\n",
            "28/28 [==============================] - 15s 532ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n",
            "Epoch 7/10\n",
            "28/28 [==============================] - 15s 538ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n",
            "Epoch 8/10\n",
            "28/28 [==============================] - 16s 556ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n",
            "Epoch 9/10\n",
            "28/28 [==============================] - 16s 570ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n",
            "Epoch 10/10\n",
            "28/28 [==============================] - 15s 532ms/step -
loss: 13.2345 - accuracy: 0.1321 - mse: 0.8679 - mae: 0.8679 - val_loss:
13.2149 - val_accuracy: 0.1334 - val_mse: 0.8666 - val_mae: 0.8666\n"
          ]
        },
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "<keras.callbacks.History at 0x7f475002f650>"
            ]
          },
          "metadata": {},
          "execution_count": 46
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
```

```
        "##**Save the Model**"
      ],
      "metadata": {
        "id": "TFgUHCLdT9zw"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "model.save(\"/assignment4_model.h5\")"
      ],
      "metadata": {
        "id": "eYgXFNPjTK7j"
      },
      "execution_count": 34,
      "outputs": []
    },
    {
      "cell_type": "markdown",
      "source": [
        "##**Test The Model**"
      ],
      "metadata": {
        "id": "mhu_IzLMUMVc"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "test_sequences = tok.texts_to_sequences(X_test)\n",
        "test_sequences_matrix =
sequence.pad_sequences(test_sequences,maxlen=max_len)\n",
        "accr = model.evaluate(test_sequences_matrix,Y_test)\n",
        "print('Test set\\n Loss: {:0.3f}\\n
Accuracy:{:0.3f}'.format(accr[0],accr[1]))"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "ZvkJLllpV5EF",
        "outputId": "817efbb0-132c-474c-fc61-03ba11e6396a"
      },
      "execution_count": 47,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "35/35 [==============================] - 5s 129ms/step -
loss: 13.0747 - accuracy: 0.1426 - mse: 0.8575 - mae: 0.8578\n",
            "Test set\n",
            " Loss: 13.075\n",
            " Accuracy:0.143\n"
          ]
        }
      ]
    },
```

```
    {
      "cell_type": "code",
      "source": [
        "from tensorflow.keras.models import load_model\n",
        "m2 = load_model(\"/assignment4_model.h5\")\n",
        "m2.evaluate(test_sequences_matrix,Y_test)"
      ],
      "metadata": {
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "id": "mcDs1LGyTUzK",
        "outputId": "d0f36367-c7bf-44d0-bf5d-0e780054c0ab"
      },
      "execution_count": 37,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "35/35 [==============================] - 3s 79ms/step -
loss: 0.0989 - accuracy: 0.9865 - mse: 0.0216 - mae: 0.1009\n"
          ]
        },
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "[0.09894020110368729,\n",
              " 0.9865471124649048,\n",
              " 0.021568873897194862,\n",
              " 0.10085303336381912]"
            ]
          },
          "metadata": {},
          "execution_count": 37
        }
      ]
    }
  ]
}
```